

Multidimensional Cube Packing*

Y. Kohayakawa[†] F.K. Miyazawa[‡] P. Raghavan[§] Y. Wakabayashi[†]

Abstract

We consider the *d-dimensional cube packing problem* (*d*-CPP): given a list L of *d*-dimensional cubes and (an unlimited quantity of) *d*-dimensional unit-capacity cubes, called *bins*, find a packing of L into the minimum number of bins. We present two approximation algorithms for *d*-CPP, for fixed d . The first algorithm has an asymptotic performance bound that can be made arbitrarily close to $2 - (1/2)^d$. The second algorithm is an improvement of the first and has an asymptotic performance bound that can be made arbitrarily close to $2 - (2/3)^d$. To our knowledge, these results improve the bounds known so far for $d = 2$ and $d = 3$, and are the first results with bounds that are not exponential in the dimension.

Key Words: Approximation algorithms, multidimensional bin packing, asymptotic performance.

1 Introduction

We consider a generalization of the one-dimensional bin packing problem, called here *d-dimensional cube packing problem* (*d*-CPP). This is the following problem. Given a list L of n *d*-dimensional

*This research was partially supported by MCT/CNPq under PRONEX project 107/97 and CNPq (Proc. 470608/01-3, 468516/00-0, 464114/00-4, 478818/03-3, 300334/93-1, 300301/98-7 and 304527/89-0).

[†]Instituto de Matemática e Estatística — Universidade de São Paulo. Rua do Matão, 1010 — 05508-090 — São Paulo-SP — Brazil, {yoshi,yw}@ime.usp.br.

[‡]Instituto de Computação — Universidade Estadual de Campinas, Caixa Postal 6176 — 13084-971 — Campinas-SP — Brazil, fkm@ic.unicamp.br.

[§]Verity, Inc., 892 Ross Drive, Sunnyvale, CA 94089, USA, pragh@verity.com.

cubes (possibly of different sizes) and d -dimensional unit-capacity cubes, called *bins*, find an orthogonal packing of L into the minimum number of bins. This problem is in fact a special case of the *d -dimensional bin packing problem* (d -BPP), in which one has to pack d -dimensional parallelepipeds into d -dimensional unit-capacity bins. Note that for $d = 1$ these problems coincide.

In 1989, Coppersmith and Raghavan [6] presented an online algorithm for d -BPP with asymptotic performance bound $(3 \cdot 2^d + 1)/4$. This algorithm, when specialized to d -CPP, has asymptotic performance bound $(3/2)^d - (3/4)^d + 1$. In 1999, Ferreira, Miyazawa and Wakabayashi [10] presented an asymptotic 1.99-approximation algorithm for 2-CPP; later, Miyazawa and Wakabayashi [16] presented a 2.67-approximation algorithm for 3-CPP. These algorithms can be generalized to an algorithm for d -CPP with asymptotic performance bound $(4/3)^d - (8/9)^d + 1$.

The most studied case is when $d = 1$. For this case there are asymptotic approximation schemes due to Karmarkar and Karp [11] and Fernandez de la Vega and Lueker [9]. For a recent survey on this case, see Coffman, Garey and Johnson [5]. For the case $d = 2$, Chung, Garey and Johnson [3] obtained in the early eighties an asymptotic 2.125-approximation algorithm. Recently, Caprara [2] obtained a 1.691-approximation algorithm. For 3-BPP Li and Cheng [14] and Csirik and van Vliet [8] designed algorithms with asymptotic performance bound 4.84. Their algorithms generalize to d -BPP giving algorithms with asymptotic performance bound close to 1.691^d . For a survey on approximation algorithms for packing problems we refer to Coffman, Garey and Johnson [4].

We present two approximation algorithms for d -CPP. The first algorithm has an asymptotic performance bound that can be made as close to $2 - (1/2)^d$ as desired. The second algorithm is an improvement of the first one and has an asymptotic performance bound that can be made as close to $2 - (2/3)^d$ as desired. For $d = 2$ and $d = 3$ the bounds are close to $14/9 \approx 1.56$ and $46/27 \approx 1.70$, respectively. To our knowledge, these results improve the bounds known so far for $d = 2$ and $d = 3$, and are the first results with bounds that are not exponential in the dimension

(see the last paragraph of this section). Recently, Seiden and van Stee [18] presented an algorithm for $d = 2$ with bound $14/9 + \epsilon$ that uses an idea similar to the one we present in this paper. It is most likely that the extended abstract [12], where we announced a comparable result, with explicit formulas for general d and full description of our algorithms, was unknown to those authors.

The remainder of this paper is organized as follows. In the next section we present the notation and some definitions. In Section 3 we describe restricted versions of d -CPP. In Section 4 we present our approximation algorithms for d -CPP.

We remark that after the submission of this paper two new results have appeared, both improving the result we show in this paper: Bansal and Sviridenko [1] and Correa and Kenyon [7] have obtained (independently) an asymptotic polynomial-time approximation scheme for d -CPP.

2 Notation and definitions

Given a cube c , the *size* of c , denoted by $s(c)$, is defined as the length of an edge of c . If L is a list of cubes, then we denote by $V(L)$ the total *volume* of the cubes in L . For a bin B , we also denote by $V(B)$ the volume of the cubes packed in B , also called the *volume occupation* of B . Throughout this paper whenever we consider a list L to be packed into unit bins we suppose that all its cubes have size at most 1.

Given a list L of cubes, and an algorithm \mathcal{A} , we denote by $\mathcal{A}(L)$ the number of bins used by algorithm \mathcal{A} when applied to L , and by $\text{OPT}(L)$ the number of bins used by an optimal packing of L . If \mathcal{P} is a packing of L , we denote the number of bins used in \mathcal{P} by $|\mathcal{P}|$. Some of our algorithms partition the input list L into sublists L_1, \dots, L_k and then apply specialized algorithms for each sublist L_i generating a partial packing \mathcal{P}_i . We denote the packing obtained by the union of these packings by $\mathcal{P}_1 \cup \dots \cup \mathcal{P}_k$. We say that an algorithm \mathcal{A} has *asymptotic performance bound* α if

there exists a constant β such that $\mathcal{A}(L) \leq \alpha \cdot \text{OPT}(L) + \beta$ for all input lists L . If $\beta = 0$, we also say that α is an *absolute performance bound* for algorithm \mathcal{A} . We note that 2-CPP cannot be approximated within $2 - \epsilon$ in the absolute sense, unless $P = NP$ (see [10], where this result is deduced from [13]). This and other negative results in terms of the absolute performance bounds make the asymptotic performance analyses of approximation algorithms for bin packing problems attractive.

3 Restricted d -CPP

Before considering the general problem, we present algorithms for restricted instances of d -CPP, to be used as subroutines.

3.1 Restricted sizes and types

First, we consider instances that admit a constant upper bound k on the number of different cube sizes and each cube size is at least ϵ . We denote the set of such instances by $\mathcal{I}_{\epsilon,k,d}$.

In what follows we show that in this case an optimal solution can be obtained in polynomial time by an enumeration process.

Given a list $L \in \mathcal{I}_{\epsilon,k,d}$, denote by $D(L) = (s_1, s_2, \dots, s_k)$ the list of different sizes in the list L and by $M(L) = (m_1, m_2, \dots, m_k)$ the list of the multiplicities m_i of the cubes with size s_i .

There are infinitely many ways of packing a list of cubes into a bin, but the number of different configurations is bounded if we consider only “canonical” packings. We say a packing \mathcal{P} is *canonical* if no cube in \mathcal{P} can be shifted to a “lower” position in any dimension, without overlapping cubes. Each configuration in a bin of a canonical packing is called a *canonical pattern*. With this in mind, we can compute all possible positions $\mathbb{P}^d(L)$ in a bin where a cube of L can be packed. We set $p(L) := \{p := b_1s_1 + b_2s_2 + \dots + b_k s_k : p \leq 1, 0 \leq b_i \leq m_i, b_i \in \mathbb{Z}\}$, and define $\mathbb{P}^d(L)$ as the set

$$p(L)^d = p(L) \times \dots \times p(L).$$

Claim 3.1 *For a given list L in $\mathcal{I}_{\epsilon,k,d}$, the cardinality of $\mathbb{P}^d(L)$ is bounded by a value that depends only on k , ϵ and d .*

Proof. Since each cube of L has size at least ϵ , we have $b_i \leq \lfloor 1/\epsilon \rfloor$ for all i . Thus we can conclude that $(\lfloor 1/\epsilon \rfloor + 1)^k$ is an upper bound on $p(L)$. Therefore, we have $|\mathbb{P}^d(L)| \leq (\lfloor 1/\epsilon \rfloor + 1)^{kd}$. \square

For $L \in \mathcal{I}_{\epsilon,k,d}$, let $\rho_{\epsilon,k,d}(L)$ denote the number of all possible canonical patterns of L . The following result is immediate.

Claim 3.2 *The value $\rho_{\epsilon,k,d}(L)$ is bounded by a value that depends only on k , ϵ and d .*

Proof. Note that the number of canonical patterns for L consisting of exactly i cubes is bounded by

$$k^i \binom{|\mathbb{P}^d(L)|}{i}.$$

Therefore,

$$\rho_{\epsilon,k,d}(L) \leq k \binom{|\mathbb{P}^d(L)|}{1} + k^2 \binom{|\mathbb{P}^d(L)|}{2} + \dots + k^{\epsilon^{-d}} \binom{|\mathbb{P}^d(L)|}{\epsilon^{-d}},$$

and the result follows. \square

Let $\mathcal{R}_{\epsilon,k,d}$ denote the algorithm that generates all possible canonical packings and chooses a packing using the smallest number of bins.

Lemma 3.3 *Let ϵ , k and d be fixed positive numbers and L an instance of d -CPP with $L \in \mathcal{I}_{\epsilon,k,d}$. Then L can be packed optimally in polynomial time by the algorithm $\mathcal{R}_{\epsilon,k,d}$.*

Proof. Suppose L has n cubes. Since we can have at most n bins of each pattern, the number of different packings is bounded by the polynomial $(n+1)^\rho$, where $\rho := \rho_{\epsilon,k,d}(L)$. \square

3.2 Restricted size

In this section, we consider instances where we only have restriction on the minimum size of a cube. We denote by $\mathcal{I}_{\epsilon,d}$ the set of all instances consisting of cubes with size greater than ϵ .

We shall make use of the linear rounding technique presented by Fernandez de la Vega and Lueker [9] to obtain an asymptotic approximation scheme $\mathcal{R}_{\epsilon,d}$ for instances in $\mathcal{I}_{\epsilon,d}$.

Given two instances X and Y for d -CPP, we write $X \preceq Y$ if there is an injection $f: X \rightarrow Y$ such that $s(c) \leq s(f(c))$ for all $c \in X$. The following claim is immediate.

Claim 3.4 *If X and Y are two instances for d -CPP with $X \preceq Y$, then $\text{OPT}(X) \leq \text{OPT}(Y)$.*

Denote by \overline{X} the instance with precisely $|X|$ cubes with size equal to the size of the largest cube in X . Clearly, $X \preceq \overline{X}$. We are now ready to describe the algorithm $\mathcal{R}_{\epsilon,d}$.

Algorithm $\mathcal{R}_{\epsilon,d}(L)$

Input: $L \in \mathcal{I}_{\epsilon,d}$.

Output: A packing of L into unit-capacity bins.

1. If L contains $< 2/\epsilon^{d+1}$ cubes, find an optimal packing exhaustively and return this packing. Otherwise, let $L = (c_1, \dots, c_n)$ be the input list sorted in non-increasing order of size.
2. Set $q := \lfloor n\epsilon^{d+1} \rfloor$ and partition L into groups G_0, G_1, \dots, G_k such that

$$G_0 \succeq G_1 \succeq \dots \succeq G_k,$$

where $|G_i| = q$ for all $i = 0, \dots, k-1$,

and $|G_k| \leq q$.

3. Let \mathcal{P}_0 be the packing obtained by placing each cube of G_0 into a different bin.

4. Let $J := \bigcup_{i=1}^k G_i$ and $\hat{J} := \bigcup_{i=1}^k \bar{G}_i$.
5. Let $\hat{\mathcal{P}}$ be the packing of \hat{J} generated by the algorithm $\mathcal{R}_{\epsilon,k,d}$; and let \mathcal{P}_1 be the packing of J corresponding to $\hat{\mathcal{P}}$.
6. Return $\mathcal{P}_0 \cup \mathcal{P}_1$.

Lemma 3.5 *For all input list $L \in \mathcal{I}_{\epsilon,d}$, we have*

$$\mathcal{R}_{\epsilon,d}(L) \leq (1 + \epsilon) \text{OPT}(L).$$

Furthermore, $\mathcal{R}_{\epsilon,d}$ is a polynomial time algorithm.

Proof. It suffices to consider the case in which $n \geq 2/\epsilon^{d+1}$. In step 2 we partition the sorted list L into $k + 1$ groups G_i each of which consists of $q = \lfloor n\epsilon^{d+1} \rfloor$ cubes, except perhaps for the last group, which may have fewer cubes. Note that $k + 1 = \lceil n/\lfloor \epsilon^{d+1}n \rfloor \rceil \leq \lceil 2/\epsilon^{d+1} \rceil$ and, therefore, the number of different cube types in list \hat{J} is bounded by a value that depends only on ϵ and d . Since $\bar{G}_i \preceq G_{i-1}$ ($i = 1, \dots, k$) we have $\hat{J} \preceq L$. From Lemma 3.3 and Claim 3.4, we have

$$|\mathcal{P}_1| = |\hat{\mathcal{P}}| = \text{OPT}(\hat{J}) \leq \text{OPT}(L). \tag{1}$$

The packing of G_0 uses q bins, therefore

$$|\mathcal{P}_0| = q = \lfloor n\epsilon^{d+1} \rfloor \leq n\epsilon^{d+1} \leq \epsilon V(L) \leq \epsilon \text{OPT}(L). \tag{2}$$

From inequalities (1) and (2), we have

$$\mathcal{R}_{\epsilon,d}(L) = |\mathcal{P}_0| + |\mathcal{P}_1| \leq \text{OPT}(L) + \epsilon \text{OPT}(L) \leq (1 + \epsilon) \text{OPT}(L),$$

as required. □

We now turn to an algorithm that will be used only in the improved version of our algorithm (see Section 4.2). This algorithm, which we call $\mathcal{R}'_{1/3,d}$, is designed for the special case in which all cubes have size greater than $1/3$ and is optimal. We observe that $\mathcal{R}'_{1/3,d}$ is a generalization for d -CPP of an algorithm presented by Ferreira, Miyazawa and Wakabayashi [10] for 2-CPP.

Algorithm $\mathcal{R}'_{1/3,d}(L)$

Input: $L \in \mathcal{I}_{1/3,d}$

Output: A packing of L into unit-capacity bins.

1. Partition the list L into the two sublists

$$L' := \{c \in L : s(c) > 1/2\} \quad \text{and} \quad L'' := L \setminus L'.$$

2. Sort the cubes in L' in non-decreasing order of their sizes. Let $L' = (c'_1, \dots, c'_{n'})$.
3. Sort the cubes in L'' in non-increasing order of their sizes. Let $L'' = (c''_1, \dots, c''_{n''})$.

We have

$$\frac{1}{3} < s(c''_{n''}) \leq \dots \leq s(c''_1) \leq \frac{1}{2} < s(c'_1) \leq \dots \leq s(c'_{n'}).$$

4. Call the subroutine $\mathcal{SR}_{1/3,d}$ with parameters (L', L'') .

Subroutine $\mathcal{SR}_{1/3,d}(L', L'')$

- (a) If $L' = \emptyset$ and $L'' = \emptyset$ then return \emptyset ;
- (b) else if $L' = \emptyset$ then generate a packing \mathcal{P} placing the first $m := \min\{2^d, n''\}$ cubes of L'' into one bin;

- (c) else if $L'' = \emptyset$ then generate a packing \mathcal{P} placing the cube c'_1 into one bin;
- (d) else if $s(c'_1) + s(c''_1) > 1$ then generate a packing \mathcal{P} placing the largest $m := \min\{2^d, n''\}$ cubes of L'' into one bin;
- (e) else generate a packing \mathcal{P} placing the cube c'_1 and the largest $m := \min\{2^d - 1, n''\}$ cubes of L'' into one bin.
- (f) Remove from L' and L'' the cubes that have been packed.
- (g) Return $\mathcal{P} \cup \mathcal{SR}_{1/3,d}(L', L'')$.

First we prove that the packing generated by the algorithm $\mathcal{R}'_{1/3,d}$ has a very simple structure. Indeed, we show that the bins generated by this algorithm, except for possibly one, may have only three different configurations.

Lemma 3.6 *Let $L \in \mathcal{I}_{1/3,d}$ be an instance for d -CPP, $L' := \{c \in L : s(c) > \frac{1}{2}\}$ and $L'' := L \setminus L'$. Then the algorithm $\mathcal{R}'_{1/3,d}$ applied to L generates a packing where each bin, except for possibly one, has one of the following configurations.*

C1 : configuration consisting of 1 cube of L' and $2^d - 1$ cubes of L'' .

C2 : configuration consisting of exactly 1 cube of L' .

C3 : configuration consisting of 2^d cubes of L'' .

Proof. Note that each call of the subroutine $\mathcal{SR}_{1/3,d}$ generates a one-bin packing and then packs the remaining cubes recursively. Let us then analyse steps (b)–(e) of this subroutine, where a packing into one bin is generated.

Clearly, the bin generated in step (c) has configuration C2. In steps (b), (d) and (e), the bin that is generated does not have one of the three configurations only when the value of m is n'' and $n'' < 2^d$ (in steps (b) and (d)) or $n'' < 2^d - 1$ (in step (e)). But this happens only once,

and after step (f) we have $L'' = \emptyset$ and therefore the next bins to be generated, if any, will have configuration $C2$. □

Theorem 3.7 *The algorithm $\mathcal{R}'_{1/3,d}$ finds an optimal solution for d -CPP restricted to instances $L \in \mathcal{I}_{1/3,d}$ in polynomial time.*

Proof. The algorithm $\mathcal{R}'_{1/3,d}$ partitions the list L into two sublists L' and L'' , sorts them and calls the subroutine $\mathcal{SR}_{1/3,d}$. In each call, this subroutine generates a one-bin packing and then packs the remaining cubes recursively.

Consider a call of the subroutine $\mathcal{SR}_{1/3,d}$ with parameters (L', L'') , and let B be the bin generated in this call. We claim that there is an optimal packing \mathcal{P}^* of $L' \cup L''$ that has a bin B^* with the same cubes as B . This is clearly true if the packing of the bin B is generated in steps (a), (b), or (c). (For step (b), it is crucial that the cubes have size greater than $1/3$.)

Now suppose that B is generated in step (d). In this case, the largest cube c''_1 of L'' cannot be packed with the smallest cube of L' . Therefore, any optimal packing \mathcal{P}^* does not have c''_1 with any other cube in L' . Let B^* be the bin of \mathcal{P}^* containing the cube c''_1 . Since any 2^d cubes of L'' can be packed together in a bin, we can exchange the cubes of B^* , so that it ends up with the cubes packed in B .

The proof for the case in which B is generated in step (e) is analogous. The result follows by induction. □

4 Approximation algorithms for d -CPP

In this section we present two approximation algorithms for d -CPP. The first has asymptotic performance bound that can be made as close to $2 - (1/2)^d$ as desired and the second has asymptotic performance bound that can be made as close to $2 - (2/3)^d$ as desired.

Both algorithms use the NFDH (Next Fit Decreasing Height) algorithm as a subroutine, to be described in what follows.

The algorithm NFDH generates t -dimensional strips (using dimensions $1, \dots, t$) using $(t - 1)$ -dimensional strips (dimensions $1, \dots, t - 1$) which are packed in non-increasing order of their size (the size of a strip is the size of the largest cube in it). The $(t - 1)$ -dimensional strips are packed side by side in dimension t until a $(t - 1)$ -dimensional strip cannot be packed (the sum of the $(t - 1)$ -dimensional strip sizes cannot be greater than 1). In this case, the $(t - 1)$ -dimensional strip starts a new t -dimensional strip. The 0-dimensional strips are the cubes themselves and the d -dimensional strips are the packings generated into bins. For more details of NFDH, see [15].

The following result was proved by Meir and Moser [15] for the algorithm NFDH. We can derive from it two corollaries which will be useful to prove the bounds for our algorithms.

Theorem 4.1 *Let L be a list of d -dimensional cubes with maximum size s . Then L can be packed by the algorithm NFDH into a d -dimensional $a_1 \times a_2 \times \dots \times a_d$ parallelepiped if*

$$V(L) \leq s^d + (a_1 - s)(a_2 - s) \cdots (a_d - s).$$

Corollary 4.2 *Let \mathcal{P} be the packing into unit bins obtained by applying the algorithm NFDH to a list L consisting of d -dimensional cubes with size at most ϵ . Then each sublist of cubes packed in a bin used by \mathcal{P} , except for possibly one, has volume greater than $(1 - \epsilon)^d$.*

Proof. Suppose \mathcal{P} uses m bins, and let $L_i = (c_1^i, \dots, c_{n_i}^i)$ be the sublist packed in the i -th bin. Since the set $L_i \cup \{c_1^{i+1}\}$, for $1 \leq i \leq m - 1$, could not be packed by the algorithm NFDH in the i -th bin, by Theorem 4.1, we have $V(L_i \cup \{c_1^{i+1}\}) > s(c_1^i)^d + (1 - s(c_1^i))^d$. Hence, for $i = 1, \dots, m - 1$, we have

$$V(L_i) > s(c_1^i)^d + (1 - s(c_1^i))^d - s(c_1^{i+1})^d \geq (1 - s(c_1^i))^d \geq (1 - \epsilon)^d.$$

□

Corollary 4.3 *Let \mathcal{P} be the packing into unit bins obtained by applying the algorithm NFDH to a list L of d -dimensional cubes. Then each sublist of cubes packed in a bin used by \mathcal{P} , except for possibly one, has volume at least $(1/2)^d$.*

Proof. Clearly, it suffices to analyse the sublists (packed in a bin) containing cubes with size at most $1/2$. Consider then a list consisting of these sublists. Applying Corollary 4.2 to this list the result is immediate. □

We now state a technical lemma [17] that will be useful in the proofs of Theorems 4.5 and 4.7.

Lemma 4.4 *Suppose a, b, γ, δ are real numbers such that $a > 0$ and $0 < \gamma < \delta < 1$. Then*

$$\frac{a + b}{\max\{a, \gamma a + \delta b\}} \leq 1 + \frac{1 - \gamma}{\delta}.$$

4.1 First Algorithm

Now we are ready to describe the first algorithm of this section, called $\mathcal{A}'_{\epsilon, d}$, which depends on a parameter ϵ . This parameter is used to subdivide the list L into two sublists: one consisting of “large” cubes (size greater than ϵ) and the other consisting of “small” cubes. For the large cubes we use the asymptotic approximation scheme presented in Section 3.2, and for the small cubes we use the algorithm NFDH. In order to obtain bounds for the volume of the large cubes we also use the algorithm NFDH.

Algorithm $\mathcal{A}'_{\epsilon, d}(L)$

Input: A list L of d -cubes.

Output: A packing of L into unit-capacity bins.

1. Partition the list L into the two sublists

$$L' := \{c \in L : s(c) > \epsilon\} \quad \text{and} \quad L'' := L \setminus L'.$$

2. Generate a packing \mathcal{P}'_1 of L' using the algorithm $\mathcal{R}_{\epsilon,d}$.
3. Generate a packing \mathcal{P}'_2 of L' using the algorithm NFDH.
4. Let \mathcal{P}' be a packing in $\{\mathcal{P}'_1, \mathcal{P}'_2\}$ that uses the least number of bins.
5. Generate a packing \mathcal{P}'' of L'' using algorithm NFDH.
6. Return $\mathcal{P}' \cup \mathcal{P}''$.

Theorem 4.5 *For fixed values of d and ϵ the algorithm $\mathcal{A}'_{\epsilon,d}$ runs in polynomial time. Furthermore, we have*

$$\mathcal{A}'_{\epsilon,d}(L) \leq \alpha_{\epsilon,d} \text{OPT}(L) + 2,$$

where $\alpha_{\epsilon,d} \rightarrow 2 - (1/2)^d$ as $\epsilon \rightarrow 0$.

Proof. Since the algorithms $\mathcal{R}_{\epsilon,d}$ and NFDH are polynomial time algorithms, $\mathcal{A}'_{\epsilon,d}$ is also a polynomial time algorithm. Let us analyse the performance of $\mathcal{A}'_{\epsilon,d}$.

Let $n' := |\mathcal{P}'| - 1$ and $n'' := |\mathcal{P}''| - 1$. From steps 3 and 4 and Corollary 4.3, we have

$$V(L') \geq \frac{1}{2^d} (|\mathcal{P}'_2| - 1) \geq \frac{1}{2^d} (|\mathcal{P}'| - 1) = \frac{1}{2^d} n'. \quad (3)$$

From step 5 and Corollary 4.2, we have

$$V(L'') \geq (1 - \epsilon)^d (|\mathcal{P}''| - 1) = (1 - \epsilon)^d n''. \quad (4)$$

From inequalities (3) and (4) and the fact that the volume of the cubes in L is a lower bound for the optimum packing, we conclude that

$$\text{OPT}(L) \geq V(L) = V(L') + V(L'') \geq n'/2^d + (1 - \epsilon)^d n''. \quad (5)$$

The packing \mathcal{P}'_1 of sublist L' is generated by an asymptotic approximation scheme (see Lemma 3.5).

Since from step 4 the packing \mathcal{P}' of L' is such that $|\mathcal{P}'| \leq |\mathcal{P}'_1|$, we have

$$\text{OPT}(L) \geq \text{OPT}(L') \geq \frac{1}{1 + \epsilon} |\mathcal{P}'| \geq \frac{n'}{1 + \epsilon}. \quad (6)$$

From inequalities (5) and (6), we have

$$\text{OPT}(L) \geq \max \left\{ n'/(1 + \epsilon), n'/2^d + (1 - \epsilon)^d n'' \right\}. \quad (7)$$

Since $\mathcal{A}'_{\epsilon,d}(L) = |\mathcal{P}'| + |\mathcal{P}''|$, we obtain

$$\mathcal{A}'_{\epsilon,d}(L) = (n' + 1) + (n'' + 1) = \frac{n' + n''}{\text{OPT}(L)} \text{OPT}(L) + 2 \leq \alpha_{\epsilon,d} \text{OPT}(L) + 2,$$

where

$$\alpha_{\epsilon,d} = \frac{n' + n''}{\max \{ n'/(1 + \epsilon), n'/2^d + (1 - \epsilon)^d n'' \}}.$$

If $n' \leq 0$ the result follows from (5). If $n' > 0$ then, rewriting $\alpha_{\epsilon,d}$ as

$$\alpha_{\epsilon,d} = (1 + \epsilon) \frac{n' + n''}{\max \{ n', ((1 + \epsilon)/2^d) n' + (1 + \epsilon)(1 - \epsilon)^d n'' \}},$$

and using Lemma 4.4, we conclude that

$$\alpha_{\epsilon,d} \leq (1 + \epsilon) + \frac{1 - (1 + \epsilon)/2^d}{(1 - \epsilon)^d}.$$

Thus, $\lim_{\epsilon \rightarrow 0} \alpha_{\epsilon,d} \leq 2 - (1/2)^d$, and the proof of the theorem is now complete. □

Proposition 4.6 *If $\epsilon \geq 1/2$ the algorithm $\mathcal{A}'_{\epsilon,d}$ finds an optimal solution. For $\epsilon < 1/2$, the asymptotic performance bound $2 - (1/2)^d$ of the algorithm $\mathcal{A}'_{\epsilon,d}$ is tight.*

Proof. Let us consider the case $\epsilon < 1/2$; for the other case the result is immediate. Let L' be a list of n' cubes of size $1/2 + \xi$, where n' is a large integer and ξ is a small real such that $\xi \leq \sqrt[d]{1/n' + 1/2^d} - 1/2$ and $1/(2\xi)$ is an integer greater than 1. Let L'' be a list of $n'' := \lceil (1 - 1/2^d)n' \rceil / \xi^d$ cubes of size ξ , and let $L := L' \cup L''$.

Consider the packing of the list L generated by the algorithm $\mathcal{A}'_{\epsilon,d}$. The algorithm $\mathcal{A}'_{\epsilon,d}$ packs L' and L'' separately: for L' it uses n' bins and for L'' it uses other $\lceil (1 - (1/2)^d)n' \rceil$ bins. That is, $\mathcal{A}'_{\epsilon,d}(L) = n' + \lceil (1 - (1/2)^d)n' \rceil$.

Now let \mathcal{P}^* be the packing obtained as follows: first pack the n' cubes of L' using the algorithm $\mathcal{R}_{\epsilon,d}$; now pack the n'' cubes of L'' into the unused part of the n' partially packed bins, plus an additional new bin. Note that $1 - \xi$ is divisible by ξ . Thus, it suffices to show that

$$n' - \left(\frac{1}{2} + \xi\right)^d + 1 \geq \left\lceil \left(1 - \frac{1}{2^d}\right) n' \right\rceil.$$

The left-hand side is the volume of the unused part of the n' bins plus the volume of an additional new bin, and the right-hand side is the volume of the n'' cubes of L'' . Using the fact that $\xi \leq \sqrt[d]{1/n' + 1/2^d} - 1/2$, it is not difficult to conclude that this inequality holds. Clearly, \mathcal{P}^* is an

optimal packing of L . Thus, $\text{OPT}(L) = n' + 1$, and hence the ratio $\mathcal{A}'_{\epsilon,d}(L)/\text{OPT}(L)$ can be made as close to $2 - (1/2)^d$ as desired.

□

4.2 Improved Algorithm

In this section we present an algorithm that is an improvement of $\mathcal{A}'_{\epsilon,d}$. This algorithm has an asymptotic performance bound that can be made as close to $2 - (2/3)^d$ as desired. We call this algorithm $\mathcal{A}_{\epsilon,d}$.

In the previous section we presented an algorithm that partitions the input list L into two sublists L' and L'' and generates a packing consisting of two parts. Part (i), for the list L' (of the cubes with size greater than ϵ), consists of an almost optimal packing but possibly with a poor bound for the volume occupation in each bin. Part (ii), for the list L'' (of the cubes with size at most ϵ), consists of a packing with a good bound for the volume occupation.

The algorithm $\mathcal{A}_{\epsilon,d}$ uses the small cubes of part (ii) to fill the bins of part (i) with poor volume occupation. Since these bins may have very complex item allocation we first reorganize the bins with very poor volume occupation in such a way as to have a packing with a more tractable configuration. After this reorganization and the packing of small items, we have one of the following two situations. *Either* we have packed all the small cubes into bins of part (i), *or else* we were not able to pack all the small cubes in the non-occupied space of the bins in part (i).

As we shall see, in the first case, we generate an almost optimal packing. In the second case, we obtain a better volume bound for the (newly generated) bins of part (i) and this leads us to an improvement of the final bound.

Let us describe the main steps of the algorithm. In step 1 we subdivide the input list into two sublists, L' and L'' , as in the previous algorithm. In step 2 we use an asymptotic approximation

scheme to obtain a packing \mathcal{P}' for the sublist L' . In step 3 we separate the bins of \mathcal{P}' with poor volume occupation and in step 4 we reorganize these bins so as to get all bins, except for perhaps one, with good volume occupation, or bins with only one large cube and poor volume occupation. In step 5 we pack the small cubes of sublist L'' into the remaining space of the bins with only one large cube.

Algorithm $\mathcal{A}_{\epsilon,d}(L)$

Input: A list L of d -cubes.

Output: A packing of L into unit-capacity bins.

1. Partition the list L into the two sublists

$$L' := \{c \in L : s(c) > \epsilon\} \quad \text{and} \quad L'' := L \setminus L'.$$

2. Generate a packing \mathcal{P}' of the sublist L' using algorithm $\mathcal{R}_{\epsilon,d}$.
3. Let \mathcal{P}'_1 consist of all bins of packing \mathcal{P}' with volume occupation less than $(2/3)^d$, and let \mathcal{P}'_2 consist of the remaining bins of \mathcal{P}' . Let L'_1 be the cubes in \mathcal{P}'_1 .
4. Generate a new packing $\widehat{\mathcal{P}}'_1$ of L'_1 as follows

4.1 Let $S_i := \{c \in L'_1 : s(c) \in \left(\frac{1}{i+1}, \frac{1}{i}\right]\}$, for $i = 1, \dots, 8$ and

let $S_9 := \{c \in L'_1 : s(c) \leq \frac{1}{9}\}$.

4.2 Generate a packing \mathcal{P}_{12} of $S_1 \cup S_2$ using the algorithm $\mathcal{R}'_{1/3,d}$ and let \mathcal{U} be the set of bins in \mathcal{P}_{12} that contain exactly one cube.

4.3 For $i = 3, \dots, 8$, do the following

4.3.1 While there remain items from S_i to be packed do the following

4.3.1.1 While \mathcal{U} is not empty

Let B be a bin in \mathcal{U} ;

If $i \in \{3, 4, 5\}$ then let $m := i^d - (i - 1)^d$ else let $m := i^d - (i - 2)^d$;

Pack (up to) m cubes of S_i (around the unique cube in B);

Update S_i ;

Remove B from \mathcal{U} .

4.3.1.2 If \mathcal{U} is empty, pack (up to) i^d cubes of S_i into a new bin B .

4.4 Pack the cubes in S_9 , as follows

4.4.1 Carve out $3^d - 2^d$ smaller bins with size $1/3$ from the empty space inside

B . Let \mathcal{U}' be the set of these $(3^d - 2^d)|\mathcal{U}|$ smaller bins.

4.4.2 Pack the cubes of S_9 into the bins of \mathcal{U}' using NFDH.

4.4.3 If any cubes of S_9 remain unpacked, pack them into new unit bins by NFDH.

5. Pack the cubes of L'' into the bins of \mathcal{U}' using NFDH.

6. If any cubes of L'' remain unpacked, pack them into new unit bins by NFDH.

7. Return the generated packing \mathcal{P} .

Theorem 4.7 *For fixed values of d and ϵ the algorithm $\mathcal{A}_{\epsilon,d}$ runs in polynomial time. Furthermore, we have*

$$\mathcal{A}_{\epsilon,d}(L) \leq \alpha_{\epsilon,d} \text{OPT}(L) + 9,$$

where $\alpha_{\epsilon,d} \rightarrow 2 - (2/3)^d$ as $\epsilon \rightarrow 0$.

Proof. From steps 2 and 3 and Lemma 3.5, we have

$$|\mathcal{P}'_1| + |\mathcal{P}'_2| = |\mathcal{P}'| \leq (1 + \epsilon) \text{OPT}(L') \leq (1 + \epsilon) \text{OPT}(L), \quad (8)$$

where \mathcal{P}'_1 consists of the bins of \mathcal{P}' with volume occupation less than $(2/3)^d$.

In what follows we shall prove the following claim: *in step 4, the packing \mathcal{P}'_1 of L'_1 is reorganized in such a way that all bins with volume occupation less than $(2/3)^d$ will end up with volume occupation at least $(2/3)^d$, except for possibly 8 bins.*

To prove the claim, we first note that in step 4.2 the algorithm $\mathcal{R}'_{1/3,d}$ generates an optimal packing \mathcal{P}_{12} of $S_1 \cup S_2$ which contains bins with configurations $C1$, $C2$ or $C3$, described in Lemma 3.6. The bins of \mathcal{P}_{12} with configuration $C1$ (containing 1 cube with volume at least $1/2^d$ and $2^d - 1$ cubes with volume at least $1/3^d$) have volume occupation at least $1/2^d + (2^d - 1)1/3^d$, which is greater than $(2/3)^d$. The bins of \mathcal{P}_{12} with configuration $C3$ (containing 2^d cubes with volume at least $1/3^d$) have volume occupation at least $(2/3)^d$.

The only bins of \mathcal{P}_{12} with volume occupation less than $(2/3)^d$ are the bins with configuration $C2$. These are exactly the bins \mathcal{U} taken in step 4.2. Note that any bin with configuration $C2$ has exactly one cube with size less than $2/3$ and therefore we have space to pack smaller cubes around it. The packing of the remaining cubes of L'_1 into these spaces is accomplished in steps 4.3 and 4.4. Moreover, we shall show that, after these steps, the only bins with volume occupation less than $(2/3)^d$ that could remain still have configuration $C2$, except perhaps for a constant number of bins.

We prove the previous statement considering the packing of each sublist S_i , $i = 3, \dots, 8$. First, let us consider the packing of the cubes in S_3 . Suppose that \mathcal{U} contains at least one bin B . In this case, we can pack at least $3^d - (3 - 1)^d$ cubes around the unique cube in B . This is possible because each cube of S_3 has size at most $1/3$ and the cube in B has size at most $2/3$. Since each cube of S_3 has volume at least $1/4^d$ and the cube in B has volume at least $1/2^d$, the bins $B \in \mathcal{U}$

after receiving the cubes of S_3 will have volume occupation

$$V(B) \geq \frac{1}{2^d} + (3^d - 2^d) \frac{1}{4^d} \geq \left(\frac{2}{3}\right)^d.$$

This holds for all such bins B , except perhaps for the last one. If during the packing of the cubes in S_3 the set \mathcal{U} becomes empty, then the algorithm packs the remaining cubes of S_3 by placing 3^d cubes in each new unit bin. In this case, each bin B of this type has volume occupation

$$V(B) \geq 3^d \frac{1}{4^d} \geq \left(\frac{2}{3}\right)^d,$$

except perhaps for the last one.

The analysis for the packing of the sublists S_4, \dots, S_8 is analogous. We can prove a volume occupation of $(2/3)^d$ for each rearranged bin of \mathcal{U} , except perhaps for one bin in each of these sublists. For the new bins we can also guarantee the same volume occupation.

Let us consider the packing of the cubes in S_9 into the remaining bins of \mathcal{U} . Suppose there is a bin B in \mathcal{U} . Since the unique cube placed (so far) in B has size at most $2/3$, we can carve out $3^d - 2^d$ smaller bins of size $1/3$ from the empty space inside B . The set of these small bins is denoted by \mathcal{U}' , and in step 4.4.2 the algorithm NFDH is used to pack the cubes of S_9 into these bins (inside B). From Corollary 4.2, the algorithm NFDH generates packings into bins $B' \in \mathcal{U}'$ with volume occupation at least $(1/3 - 1/9)^d$, except for possibly one bin. Since we have $3^d - 2^d$ bins of \mathcal{U}' inside each bin B of \mathcal{U} , the bins B after being filled with cubes of S_9 will have a volume occupation

$$V(B) \geq \frac{1}{2^d} + (3^d - 2^d) \left(\frac{1}{3} - \frac{1}{9}\right)^d \geq \frac{1}{2^d} + \frac{2^d}{3^d} - \frac{4^d}{9^d} > \left(\frac{2}{3}\right)^d.$$

If during the packing of the cubes in S_9 the set \mathcal{U}' becomes empty, then the algorithm packs the

remaining cubes of S_9 into new unit bins. From Corollary 4.2, each bin B of this type has volume occupation

$$V(B) > \left(1 - \frac{1}{9}\right)^d > \left(\frac{2}{3}\right)^d.$$

From the previous inequalities, we can conclude that, after step 4, the only bins in \mathcal{U} which have volume occupation less than $(2/3)^d$ are the remaining bins of configuration $C2$, except for possibly 8 bins (one for each sublist S_i , $i = 2, \dots, 9$). *The proof of the claim is now complete.*

At this point, we can conclude the following: If all cubes of S_3, \dots, S_9 were packed inside bins of \mathcal{U} , then the packing $\widehat{\mathcal{P}}'_1$ does not use more bins than the optimal packing \mathcal{P}_{12} , and therefore the packing $\widehat{\mathcal{P}}'_1$ is also optimal. That is

$$|\widehat{\mathcal{P}}'_1| \leq |\mathcal{P}'_1|. \tag{9}$$

If the set \mathcal{U} becomes empty in some iteration, we have obtained a packing of L'_1 with volume occupation of at least $(2/3)^d$ in each bin, except for possibly 8 bins. Since the bins of the packing \mathcal{P}'_1 have volume occupation less than $(2/3)^d$, we can conclude that the packing $\widehat{\mathcal{P}}'_1$ has at most 8 bins more than the packing \mathcal{P}'_1 . That is,

$$|\widehat{\mathcal{P}}'_1| \leq |\mathcal{P}'_1| + 8. \tag{10}$$

From inequalities (9), (10) and (8), we have

$$|\widehat{\mathcal{P}}'_1| + |\mathcal{P}'_2| \leq (1 + \epsilon)\text{OPT}(L) + 8. \tag{11}$$

At last, we have to consider the packing of the cubes in L'' generated in step 5. The cubes of this sublist is first packed by the algorithm NFDH into the remaining bins of \mathcal{U}' , which are inside

the bins of $\widehat{\mathcal{P}}_1^l$ with configuration $C2$. If necessary, new unit bins are used. The analysis of this step is divided into two cases:

Case 1. All cubes of L'' have been placed inside the bins of \mathcal{U}' . In this case, we have not used any new unit bin and, therefore,

$$|\mathcal{P}| = |\widehat{\mathcal{P}}_1^l| + |\mathcal{P}'_2| \leq (1 + \epsilon)\text{OPT}(L) + 8. \quad (12)$$

Case 2. New unit bins have been used in the packing of L'' . In this case, each bin B of \mathcal{U}' has been “filled” with cubes in L'' . From Corollary 4.2, the bin B has volume occupation

$$V(B) > (1 - \epsilon)^d > \left(\frac{2}{3}\right)^d.$$

Let $\widehat{\widehat{\mathcal{P}}}_1^l$ be the packing $\widehat{\mathcal{P}}_1^l$ with the bins of configuration $C2$ filled with cubes of L'' , and $\widehat{\mathcal{P}}''$ be the packing of the remaining cubes in L'' into new unit bins. Denote the set of cubes packed in $\widehat{\mathcal{P}}''$ as \widehat{L}'' . Since the packing $\widehat{\widehat{\mathcal{P}}}_1^l$ and the packing $\widehat{\mathcal{P}}_1^l$ have the same number of bins and the packing $\mathcal{P}'_1 \cup \mathcal{P}'_2$ is an asymptotically optimal packing, the packing $\widehat{\widehat{\mathcal{P}}}_1^l \cup \mathcal{P}'_2$ is also an asymptotically optimal packing.

Now, the final packing \mathcal{P} consists of two parts: an almost asymptotically optimal packing $\widehat{\widehat{\mathcal{P}}}_1^l \cup \mathcal{P}'_2$,

$$|\widehat{\widehat{\mathcal{P}}}_1^l \cup \mathcal{P}'_2| \leq (1 + \epsilon)\text{OPT}(L) + 8,$$

with volume occupation of $(2/3)^d$ in each bin, except for possibly 8 bins; and the packing $\widehat{\mathcal{P}}''$ containing the remaining cubes of L'' into unit bins. From Corollary 4.2, we have

$$|\widehat{\mathcal{P}}''| \leq \frac{1}{(1 - \epsilon)^d} V(\widehat{L}'') + 1.$$

Proceeding as in the proof of Theorem 4.5, we may conclude that

$$\mathcal{A}_{\epsilon,d}(L) \leq \alpha_{\epsilon,d} \text{OPT}(L) + 9,$$

where $\alpha_{\epsilon,d} = (n' + n'')/\max\{1/(1 + \epsilon)n', (2/3)^d n' + (1 - \epsilon)^d n''\}$. From Lemma 4.4, we have

$$\lim_{\epsilon \rightarrow 0} \alpha_{\epsilon,d} \leq 2 - (2/3)^d.$$

Finally, note that all steps of the algorithm $\mathcal{A}_{\epsilon,d}$ can be implemented to run in polynomial time. \square

Proposition 4.8 *If $\epsilon \geq 1/2$ the algorithm $\mathcal{A}_{\epsilon,d}$ finds an optimal solution. For $\epsilon < 1/2$, the asymptotic performance bound $2 - (2/3)^d$ of the algorithm $\mathcal{A}_{\epsilon,d}$ is tight.*

Proof. The proof of this result is similar to the proof of Proposition 4.6. Consider a list $L := L' \cup L''$, such that L' has n' cubes of size $2/3 + \xi$ and L'' has $n'' := \lceil (1 - (2/3)^d)n' \rceil / \xi^d$ cubes of size ξ , where n' is a large integer and ξ is a small real; we omit the details. \square

Acknowledgements. We are most grateful to David Johnson for many valuable comments and suggestions that improved the presentation of this paper.

References

- [1] N. Bansal and M. Sridenko. New approximability and inapproximability results for 2-dimensional bin packing. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 189–196, 2004.
- [2] A. Caprara. Packing 2-dimensional bins in harmony. In *Foundations of Computer Science*, pages 490–499, 2002.

- [3] F. R. K. Chung, M. R. Garey, and D. S. Johnson. On packing two-dimensional bins. *SIAM Journal on Algebraic and Discrete Methods*, 3:66–76, 1982.
- [4] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing - an updated survey. In G. Ausiello, M. Lucertini, and P. Serafini, editors, *Algorithms design for computer system design*, pages 49–106. Springer-Verlag, New York, 1984.
- [5] 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*, chapter 2, pages 46–93. PWS, 1997.
- [6] D. Coppersmith and P. Raghavan. Multidimensional on-line bin packing: algorithms and worst-case analysis. *Operations Research Letters*, 8(1):17–20, 1989.
- [7] J.R. Correa and C. Kenyon. Approximation schemes for multidimensional packing. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 179–188, 2004.
- [8] J. Csirik and A. van Vliet. An on-line algorithm for multidimensional bin packing. *Operations Research Letters*, 13:149–158, 1993.
- [9] W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.
- [10] C. E. Ferreira, F. K. Miyazawa, and Y. Wakabayashi. Packing of squares into squares. *Pesquisa Operacional*, 19(2):223–237, 1999.
- [11] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one dimensional bin packing problem. In *Proc. 23rd Ann. Symp. on Foundations of Computer Science*, pages 312–320, Los Angeles, 1982. IEEE Computer Society.

- [12] Y. Kohayakawa, F.K. Miyazawa, P. Raghavan, and Y. Wakabayashi. Multidimensional cube packing. In *Electronic Notes of Discrete Mathematics*, volume 7. Elsevier Science, 2001. Presented at the Brazilian Symposium on Graphs and Combinatorics.
- [13] J. Y-T. Leung, T. W. Tam, C. S. Wong, G. H. Young, and F. Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10:271–275, 1990.
- [14] K. Li and K-H. Cheng. A generalized harmonic algorithm for on-line multidimensional bin packing. TR UH-CS-90-2, University of Houston, January 1990.
- [15] A. Meir and L. Moser. On packing of squares and cubes. *J. Combinatorial Theory Ser. A*, 5:116–127, 1968.
- [16] F. K. Miyazawa and Y. Wakabayashi. Cube packing. In *Proc. 4th Latin American Theoretical INformatics.*, volume 1776 of *Lecture Notes in Computer Science*, pages 58–67, Punta del Este, Uruguay, 2000. Springer-Verlag.
- [17] F. K. Miyazawa and Y. Wakabayashi. Parametric on-line algorithms for packing rectangles and boxes. *European Journal on Operational Research*, 150:281–292, 2003.
- [18] S. S. Seiden and R. van Stee. New bounds for multi-dimensional packing. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 486–495, 2002.