Cube Packing

F. K. Miyazawa   *Y. Wakabayashi*

Relatório Técnico IC–99-25

Dezembro de 1999

Cube Packing*

F. K. Miyazawa[†]   Y. Wakabayashi[‡]

Abstract

The Cube Packing Problem (CPP) is defined as follows. Find a packing of a given list of (small) cubes into a minimum number of (larger) identical cubes. We show first that the approach introduced by Coppersmith and Raghavan for general online algorithms for packing problems leads to an online algorithm for CPP with asymptotic performance bound 3.954. Then we describe two other offline approximation algorithms for CPP: one with asymptotic performance bound 3.466 and the other with 2.669. A parametric version of this problem is defined and results on online and offline algorithms are presented. We did not find in the literature offline algorithms with asymptotic performance bounds as good as 2.669.

1 Introduction

The *Cube Packing Problem* (CPP) is defined as follows. Given a list L of n cubes (of different dimensions) and identical cubes, called *bins*, find a packing of the cubes of L into a minimum number of bins. The packings we consider are all orthogonal. That is, with respect to a fixed side of the bin, the sides of the cubes must be parallel or orthogonal to it.

CPP is a special case of the *Three-dimensional Bin Packing Problem* (3BP). In this problem the list L consists of rectangular boxes and the bins are also rectangular boxes. Here, we may assume that the bins are cubes, since otherwise we can scale the bins and the boxes in L correspondingly.

In 1989, Coppersmith and Raghavan [6] presented an online algorithm for 3BP, with asymptotic performance bound 6.25. Then, in 1992, Li and Cheng [11] presented an algorithm with asymptotic performance bound close to 4.93. Improving the latter result, Csirik and van Vliet [7], and also Li and Cheng [10] designed algorithms for 3BP with asymptotic performance bound 4.84 (the best bound known for this problem). Since CPP is a special case of 3BP, these algorithms can be used to solve it. Our aim is to show that algorithms with better asymptotic performance bounds can be designed.

Results of this kind have already been obtained for the 2-dimensional case, more precisely, for the *Square Packing Problem* (SPP). In this problem we are given a list of squares

and we are aked to pack them into a minimum number of square bins. In [6], Coppersmith and Raghavan observe that their technique leads to an online algorithm for SPP with asymptotic performance bound 2.6875. They also proved that any online algorithm for packing d-dimensional squares, $d \geq 2$, must have asymptotic performance bound at least $4/3$. Ferreira, Miyazawa and Wakabayashi [9] presented an offline algorithm for SPP with asymptotic performance bound 1.988. For the more general version of the 2-dimensional case, where the items of L are rectangles (instead of squares), Chung, Garey and Johnson [2] designed an algorithm with asymptotic performance bound 2.125.

For more results on packing problems the reader is referred to [1, 3, 4, 5, 8].

The remainder of this paper is organized as follows. In Section 2 we present some notation and definitions. In Section 3 we describe an online algorithm for CPP that uses an approach introduced by Coppersmith and Raghavan [6], showing that its asymptotic performance bound is at most 3.954. In Section 4 we present an offline algorithm with asymptotic performance bound 3.466. We mention a parametric version for these algorithms and derive asymptotic performance bounds. In Section 5 we present an improved version of the offline algorithm described in Section 4. We show that this algorithm has asymptotic performance bound 2.669. Finally, in Section 6 we present some concluding remarks.

2   Notation and definitions

The reader is referred to [14] for the basic concepts and terms related to packing. Without loss of generality, we assume that the bins have unit dimensions, since otherwise we can scale the cubes of the instance to fulfill this condition.

A rectangular box b with length x, width y and height z is denoted by a triplet $b = (x, y, z)$. Thus, a cube is simply a triplet of the form (x, x, x). The *size* of a cube $c = (x, x, x)$, denoted by $s(c)$, is x. Here we assume that every cube in the input list L has size at most 1. The *volume* of a list L, denoted by $V(L)$, is the sum of the volumes of the items in L.

For a given list L and algorithm $\mathcal{A}$, we denote by $\mathcal{A}(L)$ the number of bins used when algorithm $\mathcal{A}$ is applied to list L, and by $\mathrm{OPT}(L)$ the optimum number of bins for a packing of L. We say that an algorithm $\mathcal{A}$ has an *asymptotic performance bound* α if there exists a constant β such that

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

If $\beta = 0$ then we say that α is an *absolute performance bound* for algorithm $\mathcal{A}$.

If $\mathcal{P}$ is a packing, then we denote by $\#(\mathcal{P})$ the number of bins used in $\mathcal{P}$.

An algorithm to pack a list of items $L = (c_1, \ldots, c_n)$ is said to be *online* if it packs the items in the order given by the list L, without knowledge of the subsequent items on the list. An algorithm that is not online is said to be *offline*.

We consider here a *parametric* version of CPP, denoted by CPP_m, where m is a natural number. In this problem, the instance L consists of cubes with size at most $1/m$. Thus CPP_1 and CPP are the same problem.

3 The online algorithm of Coppersmith and Raghavan

In 1989, Coppersmith and Raghavan [6] introduced an online algorithm for the multidimensional bin packing problem. In this section we describe a specialized version of this algorithm for CPP. Our aim is to derive an asymptotic performance bound for this algorithm (not explicitly given in the above paper).

The main idea of the algorithm is to round up the dimensions of the items in L using a rounding set $S = \{1 = s_0, s_1, \ldots, s_i, \ldots\}$, $s_i > s_{i+1}$. The first step consists in rounding up each item size to the nearest value in S. The rounding set S for CPP is $S := S_1 \cup S_2 \cup S_3$, where

$$S_1 = \{1\}, \quad S_2 = \{1/2, 1/4, \ldots, 1/2^k, \ldots\}, \quad S_3 = \{1/3, 1/6, \ldots, 1/(3 \cdot 2^k), \ldots\}.$$

Let $\overline{x}$ be the value obtained by rounding up x to the nearest value in S. Given a cube $c = (x, x, x)$, define $\overline{c}$ as the cube $\overline{c} := (\overline{x}, \overline{x}, \overline{x})$. Let $\overline{L}$ be the list obtained from L by rounding up the sizes of the cubes to the values in the rounding set S. The idea is to pack the cubes of the list $\overline{L}$ instead of L, so that the packing of each cube $\overline{c} \in \overline{L}$ represents the packing of $c \in L$. The packing of $\overline{L}$ is generated into bins belonging to three different groups: G_1, G_2 and G_3. Each group G_i contains only bins of dimensions $(x, x, 1)$, $x \in S_i$, $i = 1, 2, 3$. A bin of dimension $(x, x, 1)$ will have only cubes $c = (x, x, x)$ packed into it. We say that a cube $c = (x, x, x)$ is of *type i*, if $\overline{x} \in S_i$, $i = 1, 2, 3$.

To pack the next unpacked cube $\overline{c} \in \overline{L}$ with size $x \in S_i$, we proceed as follows.

1. Let $B \in G_i$ be the first bin $B = (x, x, 1)$, such that $\sum_{b \in B} s(b) + x \leq 1$ (if there exists such a bin B).

2. If there is a bin B as in step 1, pack $\overline{c}$ in a Next Fit manner into B.

3. Otherwise,

   (a) take the first empty bin $C = (y, y, 1)$, $y \in S_i$, with $y > x$ and y as small as possible. If there is no such bin C, take a new bin $(1, 1, 1)$ and replace it by i^2 bins of dimensions $(\frac{1}{i}, \frac{1}{i}, 1)$ and let $C = (y, y, 1)$ be the first of these i^2 bins.

   (b) If $y > x$, then replace C by other four bins of dimensions $(\frac{y}{2}, \frac{y}{2}, 1)$. Continue in this manner replacing one of these new bins by four bins, until there is a bin C of dimension $C = (\frac{y}{2^m}, \frac{y}{2^m}, 1)$ with $\frac{y}{2^m} = x$.

   (c) Pack $\overline{c}$ in a Next Fit manner into the first bin C.

4. Update the group G_i.

Let us now analyse the asymptotic performance of the algorithm we have described. Consider $\mathcal{P}$ the packing of L generated by this algorithm, L_i the set of all cubes of type i in L, and $\mathcal{P}_i$ the set of bins of $\mathcal{P}$ having only cubes of type i. Now, let us consider the bins $B = (x, x, 1)$, $x \in S_i$, and compute the volume occupied by the cubes of $\overline{L}$ that were packed into these bins. All bins in the group G_1 are completely filled. Thus, $\#(\mathcal{P}_1) = V(\overline{L}_1)$. For the bins in the groups G_2 and G_3 the unoccupied volume is at most 1 for each group. Therefore, we have $\#(\mathcal{P}_i) \leq V(\overline{L}_i) + 1$, $i = 2, 3$.

Now, let us consider the volume we increased because of the rounding process. Each cube $c \in L_1$ has volume at least $\frac{1}{8}$ of $\overline{c}$, and each cube $c \in L_2 \cup L_3$ has volume at least $\frac{8}{27}$ of $\overline{c}$. Hence, we have the following inequalities:

$$\#(\mathcal{P}_1) \leq \frac{1}{1/8} V(L_1), \text{ and } \#(\mathcal{P}_2 \cup \mathcal{P}_3) \leq \frac{1}{8/27} V(L_2 \cup L_3) + 2.$$

Let $n_1 := \#(\mathcal{P}_1)$ and $n_{23} := \#(\mathcal{P}_2 \cup \mathcal{P}_3) - 2$. Thus, using the inequalities above and the fact that the volume of the cubes in L is a lower bound for the optimum packing, we have $\mathrm{OPT}(L) \geq V(L) \geq \frac{1}{8} n_1 + \frac{8}{27} n_{23}$.

Since $\mathrm{OPT}(L) \geq n_1$, it follows that $\mathrm{OPT}(L) \geq \max\{n_1, \frac{1}{8} n_1 + \frac{8}{27} n_{23}\}$. Now using the fact that $\#(\mathcal{P}) = \#(\mathcal{P}_1) + \#(\mathcal{P}_2 \cup \mathcal{P}_3) = n_1 + n_{23} + 2$, we have

$$\#(\mathcal{P}) \leq \alpha \cdot \mathrm{OPT}(L) + 2,$$

where $\alpha = (n_1 + n_{23})/(\max\{n_1, \frac{1}{8} n_1 + \frac{8}{27} n_{23}\})$. Analysing the two possible cases for the denominator, we obtain $\alpha \leq 3.954$.

The approach used above can also be used to develop online algorithms for the parametric version CPP_m. In this case we partition the packing into two parts. One part is an optimum packing with all bins, except perhaps the last (say n' bins), filled with m^3 cubes of volume at least $(1/(m+1))^3$ each. The other part is a packing with all bins, except perhaps a fixed number of them (say n'' bins), having an occupied volume of at least $((m+1)/(m+2))^3$.

It is not difficult to show that the asymptotic performance bound α_m of CPP_m is bounded by $(n' + n'')/(\max\{n', (m/(m+1))^3 n' + ((m+1)/(m+2))^3 n''\})$. For $m = 2$ and $m = 3$ these values are at most 2.668039 and 2.129151, respectively.

4   An offline algorithm

Before we present our first offline algorithm for CPP, let us describe the algorithm NFDH (Next Fit Decreasing Height), which is used as a subroutine.

NFDH first sorts the cubes of L in nonincreasing order of their size, say $c_1, c_2, \ldots, c_n$. The first cube c_1 is packed in the position $(0, 0, 0)$, the next one is packed in the position $(s(c_1), 0, 0)$ and so on, side by side, until a cube that does not fit in this layer is found. At this moment the next cube c_k is packed in the position $(0, s(c_1), 0)$. The process continues in this way, layer by layer, until a cube that does not fit in the first level is found. Then the algorithm packs this cube in a new level at height $s(c_1)$. When a cube cannot be packed in a bin, it is packed in a new bin. The algorithm proceeds in this way until all cubes of L have been packed.

The following results will be used in the sequel. The proof of Lemma 1 is left to the reader. Theorem 2 follows immediately from Lemma 1.

Theorem 4.1 (Meir and Moser [12]) *Any list L of k-dimensional cubes, with sizes $x_1 \geq x_2 \geq \cdots \geq x_n \geq \cdots$, can be packed by algorithm NFDH into only one k-dimensional rectangular parallelepiped of volume $a_1 \times a_2 \times \ldots \times a_k$ if $a_j > x_1$ $(j = 1, \ldots, k)$ and $x_1^k + (a_1 - x_1)(a_2 - x_1) \cdots (a_k - x_1) \geq V(L)$.*

Lemma 4.2 *For any list of cubes $L = (c_1, \ldots, c_n)$ such that $x(c_i) \leq \frac{1}{m}$, the following holds for the packing of L into unit bins:*

$$\mathrm{NFDH}(L) \leq ((m+1)/m)^3 \, V(L) + 2.$$

Theorem 4.3 *For any list of cubes $L = (b_1, \ldots, b_n)$ such that $x(b_i) \leq \frac{1}{m}$, the following holds for the packing of L into unit bins:*

$$\mathrm{NFDH}(L) \leq ((m+1)/m)^3 \, \mathrm{OPT}(L) + 2.$$

Before presenting the first offline algorithm, called CUBE, let us introduce a definition and the main ideas behind it.

If a packing $\mathcal{P}$ of a list L satisfies the inequality $\#(\mathcal{P}) \leq \frac{V(L)}{v} + C$, where v and C are constants, then we say that v is a *volume guarantee* of the packing $\mathcal{P}$ (for the list L). Algorithm CUBE uses an approach, which we call *critical set combination* (see [13]), based on the following observation.

Recall that in the analysis of the performance of the algorithm presented in Section 3 we considered the packing divided into two parts. One optimum packing, of the list L_1, with a volume guarantee $\frac{1}{8}$, and the other part, of the list $L_{23} = L_2 \cup L_3$, with a volume guarantee $\frac{8}{27}$. If we consider this partition of L, the volume we can guarantee in each bin is the best possible, as we can have cubes in L_1 with volume very close to $\frac{1}{8}$, and cubes in L_{23} for which we have a packing with volume occupation in each bin very close to $\frac{8}{27}$. In the critical set combination approach, we first define some subsets of cubes in L_1 and L_{23} with small volumes as the critical sets. Then we combine the cubes in these critical sets obtaining a partial packing that is part of an optimum packing and has volume occupation in each bin better than $\frac{1}{8}$. That is, sets of cubes that would lead to small volume occupation are set aside and they are combined appropriately so that the resulting packing has a better volume guarantee.

Theorem 4.4 *For any list L of cubes for* CPP, *we have*

$$\mathrm{CUBE}(L) \leq 3.466 \cdot \mathrm{OPT}(L) + 4.$$

Proof. First, consider the packing $\mathcal{P}_{AB}$. Since each bin of $\mathcal{P}_{AB}$, except perhaps the last, contains one cube of L_A and seven cubes of L_B, we have

$$\#(\mathcal{P}_{AB}) \leq \frac{1}{83/216} V(L_{AB}) + 1, \tag{1}$$

where L_{AB} is the set of cubes of L packed in $\mathcal{P}_{AB}$.

Now consider a partition of $\mathcal{P}'$ into three partial packings $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$, defined as follows. The packing $\mathcal{P}_1$ has the bins of $\mathcal{P}'$ with at least one cube of size greater than $\frac{1}{2}$. The packing $\mathcal{P}_2$ has the bins of $\mathcal{P}' \setminus \mathcal{P}_1$ with at least one cube of size greater than $\frac{1}{3}$. The packing $\mathcal{P}_3$ has the remaining bins, *i.e.*, the bins in $\mathcal{P}' \setminus (\mathcal{P}_1 \cup \mathcal{P}_2)$. Let L_i be the set of cubes packed in $\mathcal{P}_i$, $i = 1, 2, 3$.

Algorithm CUBE

    // To pack a list of cubes L into unit bins $B = (1, 1, 1)$.

1 Let $p = 0.354014$; and L_A, L_B be sublists of L defined as follows.

$$L_A \leftarrow \{c \in L_1 : \tfrac{1}{2} < s(c) \le (1-p)\}, \qquad L_B \leftarrow \{c \in L_2 : \tfrac{1}{3} < s(c) \le p\}.$$

2 Generate a partial packing $\mathcal{P}_{AB}$ of $L_A \cup L_B$, such that $\mathcal{P}_{AB}$ is the union of packings $\mathcal{P}^1_{AB}, \dots, \mathcal{P}^k_{AB}$, where $\mathcal{P}^i_{AB}$ is a packing generated for one bin, consisting of one cube of L_A and seven cubes of L_B, except perhaps the last (that can have fewer cubes of L_B). [The packing $\mathcal{P}_{AB}$ will contain all cubes of L_A or all cubes of L_B.]

    Update the list L by removing the cubes packed in $\mathcal{P}_{AB}$.

3 $\mathcal{P}' \leftarrow \mathrm{NFDH}(L)$;

4 Return $\mathcal{P}' \cup \mathcal{P}_{AB}$.

end algorithm.

Since all cubes of L_3 have size at most $1/3$, and they are packed in $\mathcal{P}_3$ with algorithm NFDH, by Lemma 4.2, we have

$$\#(\mathcal{P}_3) \le \frac{1}{27/64} V(L_3) + 2. \tag{2}$$

Case 1. L_B is totally packed in $\mathcal{P}_{AB}$.

    In this case, every cube of L_1 has volume at least $\frac{1}{8}$. Therefore

$$\#(\mathcal{P}_1) \;\le\; \frac{1}{1/8} V(L_1). \tag{3}$$

Now, since every cube of L_2 has size at least p and each bin of packing $\mathcal{P}_2$ has at least 8 cubes of L_2, we have

$$\#(\mathcal{P}_2) \;\le\; \frac{1}{8p^3} V(L_2) + 1. \tag{4}$$

Since $8p^3 = \min\{8p^3, \frac{83}{216}, \frac{27}{64}\}$, using (1), (2) and (4), and setting $\mathcal{P}_{aux} := \mathcal{P}_{AB} \cup \mathcal{P}_3 \cup \mathcal{P}_2$, we have

$$\#(\mathcal{P}_{aux}) \;\le\; \frac{1}{8p^3} V(L_{aux}) + 4. \tag{5}$$

Clearly, $\mathcal{P}_1$ is an optimum packing of L_1, and hence

$$\#(\mathcal{P}_1) \;\le\; \mathrm{OPT}(L). \tag{6}$$

Defining $h_1 := \#(\mathcal{P}_1)$ and $h_2 := \#(\mathcal{P}_{aux}) - 4$, and using inequalities (3), (5) and (6) we have

$$\mathrm{CUBE}(L) \le \alpha' \cdot \mathrm{OPT}(L) + 4,$$

where $\alpha' = (h_1 + h_2)/(\max\{h_1, \frac{1}{8}h_1 + 8p^3 h_2\}) \le 3.466$.

Case 2. L_A is totally packed in $\mathcal{P}_{AB}$.

In this case, the volume guarantee for the cubes in L_1 is better than the one obtained in Case 1. Each cube of L_1 has size at least $1 - p$. Thus, $\#(\mathcal{P}_1) \le \frac{1}{(1-p)^3}V(L_1)$. For the packing $\mathcal{P}_2$, we obtain a volume guarantee of at least $\frac{8}{27}$, and the same holds for the packings $\mathcal{P}_3$ and $\mathcal{P}_{AB}$. Thus, for $\mathcal{P}_{aux}$ as above, $\#(\mathcal{P}_{aux}) \le \frac{1}{8/27}V(L_{aux}) + 4$.

Since $\#(\mathcal{P}_1) \le \mathrm{OPT}(L)$, combining the previous inequalities and proceeding as in Case 1, we have

$$\mathrm{CUBE}(L) \le \alpha'' \cdot \mathrm{OPT}(L) + 4,$$

where $\alpha'' = (h_1 + h_2)/(\max\{h_1, (1 - p)^3 h_1 + \frac{8}{27}h_2\}) \le 3.466$.

The proof of the theorem follows from the results obtained in Case 1 and Case 2. We observe that the value of p was obtained by imposing equality for the values of α' and α''. $\square$

Algorithm CUBE can also be generalized for the parametric problem CPP_m. The idea is the same as the one used in algorithm CUBE. The input list is first subdivided into two parts, P_1 and P_2. Part P_1 consists of those cubes with size in $\left(\frac{1}{m+1}, \frac{1}{m}\right]$, and part P_2 consists of the remaining cubes. The critical cubes in each part are defined using an appropriate value of $p = p(m)$, and then combined. The analysis is also divided into two parts, according to which critical set is totally packed in the combined packing. It is not difficult to derive the bounds $\alpha(\mathrm{CUBE}_m)$ that can be obtained for the corresponding algorithms. For $m = 2$ and $m = 3$ the values of $\alpha(\mathrm{CUBE}_m)$ are at most 2.42362852 ($p = 0.26355815$) and 1.98710756 ($p = 0.20916664$), respectively.

5   An improved algorithm for CPP

We present in this section an algorithm for the cube packing problem that is an improvement of algorithm CUBE described in the previous section. For that, we consider another restricted version of CPP, denoted by CPP^k, where k is an integer greater than 2. In this problem the instance is a list L consisting of cubes of size greater than $\frac{1}{k}$. We use in the sequel the following result for CPP^3.

Lemma 5.1 *There is a polynomial time algorithm to solve* CPP^3.

Proof. Let $L_1 = \{c \in L : s(c) > \frac{1}{2}\}$ and $L_2 = L \setminus L_1$. Without loss of generallity, consider $L_1 = (c_1, \dots, c_k)$. Pack each cube $c_i \in L_1$ in a unit bin C_i at the corner $(0, 0, 0)$. Note that it is possible to pack seven cubes with size at most $1 - s(c_i)$ in each bin C_i. Now, for each bin C_i, consider seven other smaller bins $C_i^{(j)}$, $j = 1, \dots, 7$, each with size $1 - s(c_i)$.

Consider a bipartite graph G with vertex set $X \cup Y$, where X is the set of the small bins, and Y is precisely L_2. In G there is an edge from a cube $c \in L_2$ to a bin $C_i^{(j)} \in X$ if and only if c can be packed into $C_i^{(j)}$. Clearly, a maximum matching in G corresponds to a maximum packing of the cubes of L_2 into the bins occupied by the cubes of L_1. Denote by $\mathcal{P}_{12}$ the packing of L_1 combined with the cubes of L_2 packed with the matching strategy. The optimum packing of L can be obtained by adding to the packing $\mathcal{P}_{12}$ the bins packed with the remaining cubes of L_2 (if existent), each with 8 cubes, except perhaps the last. ☐☐

We say that a cube c is of type G, resp. M, if $s(c) \in \left(\frac{1}{2}, 1\right]$, resp. $s(c) \in \left(\frac{1}{3}, \frac{1}{2}\right]$.

Lemma 5.2 *It is possible to generate an optimum packing of an instance of CPP^3 such that each bin, except perhaps one, has one of the following configurations:*
(a) C1: configuration consisting of 1 cube of type G and 7 cubes of type M;
(b) C2: configuration consisting of exactly 1 cube of type G; and
(c) C3: configuration consisting of 8 cubes of type M.

Lemma 5.1 shows the existence of a polynomial time optimum algorithm for CPP^3. In fact, it is not difficult to design a greedy-like algorithm to solve CPP^3 in time $O(n \log n)$. Such an algorithm is given in [9] for SPP^3 (defined analogously with respect to SPP).

We are now ready to present the improved algorithm for the cube packing problem, which we call ICUBE (Improved CUBE).

Algorithm ICUBE

// To pack a list of cubes L into unit bins $B = (1, 1, 1)$.

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

2. Generate an optimum packing $\mathcal{P}_1'$ of L_1' (in polynomial time), with bins as in Lemma 5.2. That is, solve CPP^3 with input list L_1'.

3. Let $\mathcal{P}_A$ be the set of bins $B \in \mathcal{P}_1'$ having configuration $C2$ with a cube $q \in B$ with $s(q) \leq \frac{2}{3}$; let L_A be the set of cubes packed in $\mathcal{P}_A$.

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

5. Generate a packing $\mathcal{P}_{AB}$ filling the bins in $\mathcal{P}_A$ with cubes of L_B (see below).

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

7. Let $\mathcal{P}_2$ be the packing of the unpacked cubes of L_B generated by NFDH.

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

end algorithm

To generate the packing $\mathcal{P}_{AB}$, in step 5 of algorithm ICUBE, we first partition the list L_B into 5 lists, $L_{B,3}$, $L_{B,4}$, $L_{B,5}$, $L_{B,6}$, $L_{B,7}$, defined as follows. $L_{B,i} = \{c \in L_B : \frac{1}{i+1} < s(c) \leq \frac{1}{i}\}$, $i = 3, \ldots, 6$ and $L_{B,7} = \{c \in L_B : s(c) \leq \frac{1}{7}\}$. Then we combine the cubes in each of these lists with the packing $\mathcal{P}_A$ generated in step 3.

Now consider the packing of cubes of $L_{B,3}$ into bins of $\mathcal{P}_A$. Since we can pack 19 cubes of $L_{B,3}$ into each of these bins, we generate such a packing until all cubes of $L_{B,3}$ have been packed, or until there are no more bins in $\mathcal{P}_A$. We generate similar packings combining the remaining bins of $\mathcal{P}_A$ with the lists $L_{B,4}$, $L_{B,5}$ and $L_{B,6}$. To pack the cubes of $L_{B,7}$ into bins of $\mathcal{P}_A$ we consider the empty space of the bin divided into three smaller bins of dimensions $(1, 1, \frac{1}{3})$, $(1, \frac{1}{3}, \frac{1}{3})$ and $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. Then use NFDH to pack the cubes in $L_{B,7}$ into these smaller bins. We continue the packing of $L_{B,7}$ using other bins of $\mathcal{P}_A$ until there are no more unpacked cubes of $L_{B,7}$, or all bins of $\mathcal{P}_A$ have been considered.

Theorem 5.3 *For any instance L of* CPP, *we have*

$$\text{ICUBE}(L) \leq 2.669 \cdot \text{OPT}(L) + 7.$$

Proof. (Sketch) Let $\mathcal{C}_1'$, $\mathcal{C}_2'$ and $\mathcal{C}_3'$ be the set of bins used in $\mathcal{P}_1'$ with configurations C1, C2 and C3, respectively. Considering the volume guarantees of $\mathcal{C}_1'$, $\mathcal{C}_2'$ and $\mathcal{C}_3'$, we have $\#(\mathcal{C}_1') \leq \frac{1}{1/8 + 7/27} V(\mathcal{C}_1') + 1$, $\#(\mathcal{C}_2') \leq \frac{1}{1/8} V(\mathcal{C}_2') + 1$, and $\#(\mathcal{C}_3') \leq \frac{1}{8/27} V(\mathcal{C}_3') + 1$.

We call L_A the set of cubes packed in $\mathcal{C}_2'$, and consider it a *critical set* ($L_A := \{q \in L : \frac{1}{2} < s(q) \leq \frac{2}{3}\}$). The bins of $\mathcal{C}_2'$ are additionally filled with the cubes in L_1', defined as L_B, until possibly all cubes of L_B have been packed ($L_B := \{q \in L : 0 < s(q) \leq \frac{1}{3}\}$). We have two cases to analyse.

Case 1: All cubes of L_B have been packed in $\mathcal{P}_{AB}$.

The analysis of this case is simple and will be omitted.

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

Note that the volume occupation in each bin with configuration C1 or C3 is at least $\frac{8}{27}$. For the bins with configuration C2, we have a volume occupation of $\frac{1}{8}$. In step 5, the bins with configuration C2 are additionaly filled with cubes of L_B generating a combined packing $\mathcal{P}_{AB}$.

In this case, all cubes of L_A have been packed with cubes of L_B. Thus, each bin of $\mathcal{P}_{AB}$ has a volume ocupation of at least $\frac{8}{27}$. The reader can verify this fact by adding up the volume of these cubes in L_A and the cubes of $L_{B,i}$, $i = 3, \ldots, 6$. For bins combining cubes of L_A with $L_{B,7}$, we use Theorem 1 to guarantee this minimum volume ocupation for the resulting packed bins. Therefore, we have an optimum packing of L_1 with volume guarantee at least $\frac{8}{27}$. Thus we have $\#(\mathcal{P}_1) \leq \text{OPT}(L)$, and $\#(\mathcal{P}_1) \leq \frac{V(L)}{8/27} + 6$.

The packing $\mathcal{P}_2$ is generated by algorithm NFDH for a list of cubes with size not greater than $\frac{1}{3}$. Therefore, by Lemma 4.2, we have $\#(\mathcal{P}_2) \leq \frac{V(L)}{27/64} + 2$.

Now, proceeding as in the proof of Theorem 4.4, we obtain

$$\text{ICUBE}(L) \leq \alpha \cdot \text{OPT}(L) + 8,$$

where $\alpha = \frac{1945}{729} \leq 2.669$. $\square$ □

6 Concluding remarks

We have described an online algorithm for CPP that is a specialization of an approach introduced by Coppersmith and Raghavan [6] for a more general setting. Our motivation

in doing so was to obtain the asymptotic performance bound (3.954) of this algorithm, so that we could compare it with the bounds of the offline algorithms presented here.

We have shown a simple offline algorithm for CPP with asymptotic performance bound 3.466. Then we have designed another offline algorithm that is an improvement of this algorithm, with asymptotic performance bound 2.669. This result can be generalized to k-dimensional cube packing, for $k > 3$, by making use of the Theorem 4.1 and generalizing the techniques used in this paper. Both algorithms can be implemented to run in time $O(n \log n)$, where n is the number of cubes in the list L. We have also shown that if the instance consists of cubes with size greater than 1/3 there is a polynomial exact algorithm.

We did not find in the literature offlines algorithms for CPP with asymptotic performance bound as good as 2.669.

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 – an updated survey. In G. Ausiello et al. (eds.) *Algorithms design for computer system design*, 49–106. Springer-Verlag, New York, 1984.

[4] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing – a survey. In D. Hochbaum (ed.) *Approximation algorithms for NP-hard problems*, 46–93, PWS, 1997.

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

[7] J. Csirik and A. van Vliet. An on-line algorithm for multidimensional bin packing. *Operations Research Letters*, 13:149–158, 1993.

[8] H. Dyckhoff, G. Scheithauer, and J. Terno. Cutting and packing. In F. Maffioli, M. Dell'Amico and S. Martello (eds.) *Annotated Bibliographies in Combinatorial Optimization*, Chapter 22, 393–412. John Wiley, 1997.

[9] C. E. Ferreira, F. K. Miyazawa, and Y. Wakabayashi. Packing squares into squares. *Pesquisa Operacional*, a special volume on cutting and packing. To appear.

[10] 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.

[11] K. Li and K-H. Cheng. Generalized first-fit algorithms in two and three dimensions. *Int. J. Found. Comput Sci.*, 1(2):131–150, 1992.

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

[13] F. K. Miyazawa and Y. Wakabayashi. Approximation algorithms for the orthogonal z-oriented three-dimensional packing problem. To appear in *SIAM J. Computing*.

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