Parametric On-Line Packing[§]

F. K. Miyazawa[†] Y. Wakabayashi[‡]

Abstract. We present approximation algorithms for the following problems: the *on-line strip packing*, the *on-line two-dimensional bin packing*, the *on-line three-dimensional packing problem* and the *on-line container packing problem*. We consider the special case in which the items to be packed are small and must be packed on-line. We say an item is *small* if each of its dimension is at most $\frac{1}{m}$ of the respective dimension of the recipient, where m is an integer greater than 1. We analyse the asymptotic performance bound of these algorithms. To our knowledge, the bounds presented here are the best known for this special case.

Keywords: On-line packing, parametric packing, approximation algorithms.

1 Introduction

We present approximation algorithms for packing problems where the items to be packed are small compared to the dimension of the recipient. Given a positive integer m, $m \geq 2$, we define an item as being *small* if each of its dimension is at most $1/m$ of the respective dimension of the recipient. An algorithm to pack this kind of items is also said to be a *parametric algorithm*, due to the parameter m. We consider here on-line algorithms. Given a list $L = (e_1, e_2, \ldots, e_n)$ of items to be packed, an algorithm $\mathcal{A}$ to pack L is said to be *on-line* if (i) $\mathcal{A}$ packs items in the given order of L; (ii) $\mathcal{A}$ packs each item e_i without knowledge of any item e_j, $j > i$; and (iii) $\mathcal{A}$ never move an item already packed. An algorithm that is not on-line is called *off-line*. A packing problem for which one is looking for an on-line or off-line algorithm is called correspondingly an on-line or off-line problem.

The problems we consider have many applications, in special in job scheduling. One such problem, that can be formulated as a packing problem [12, 14, 15], is to find an assignment of jobs to resources (*e.g.* memory or processors), where each job uses a small amount of resource during a continuous interval of time. The objective is to minimize the total time to execute all jobs. On-line algorithms are very important in these applications, as most of the time, the operating system does not know about further jobs users can start. Moreover, operating systems must schedule jobs very fast. Therefore, the use of exact algorithms turn out to be infeasible to these applications, as the instances are large and they need a large amount of time to find a solution.

We consider the following problems:

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

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

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

1. *Strip Packing Problem* (SPP): Given a list L of rectangles, and a rectangle of width 1 and infinite height $R = (1, \infty)$, pack the rectangles of L into R, such that the height of the packing is minimized;

2. *Two-dimensional Bin Packing Problem* (2BP): Given a list L of rectangles, and rectangles of unit dimensions $R = (1, 1)$, called bins, pack the rectangles of L into a minimum number of bins;

3. *Three-dimensional Packing Problem* (TPP): Given a list L of boxes, and a box of width 1, length 1 and infinite height $B = (1, 1, \infty)$, pack the boxes of L into B such that the height of the packing is minimized;

4. *Container Packing Problem* (3BP): Given a list L of boxes, and boxes of unit dimensions $B = (1, 1, 1)$, called containers, pack the boxes of L into a minimum number of containers.

The approach used here to tackle these problems is to develop approximation algorithms: polynomial time fast algorithms that guarantee a certain quality of the solution found, compared to the optimum packing. Given an algorithm $\mathcal{A}$ for one of the problems above and a list of items L for the respective problem, we denote by $\mathcal{A}(L)$ the height or the number of recipients (depending on which problem is considered) of the packing generated by the algorithm $\mathcal{A}$ applied to the list L. We denote by $\mathrm{OPT}(L)$ the corresponding value of an optimum packing. We say that an algorithm $\mathcal{A}$ has asymptotic performance bound α if there exists a constant β such that for any instance L we have $\mathcal{A}(L) \leq \alpha \cdot \mathrm{OPT}(L) + \beta$.

Off-line algorithms for these problems have been more investigated than on-line algorithms. In the early seventies, Johnson [7, 8] presented several off-line approximation algorithms. More recently, Csirik [4] proved that the First Fit Decreasing (FFD) algorithm has asymptotic performance bound $(m+3)/(m+2) - 1/(m(m+1)(m+2))$, when m is odd, and $(m+3)/(m+2) - 2/(m(m+1)(m+2))$, when m is even, $m \geq 5$. In [2] Coffman *et al.* presented an algorithm with asymptotic performance bound $(m+2)/(m+1)$ for the *strip packing problem*. In [11] Li and Cheng presented an algorithm for the *three-dimensional packing problem* with asymptotic performance bound $(m+1)/(m-1)$, $m \geq 2$. In [16] we improve the latter bounds for the off-line two-dimensional and three-dimensional case with asymptotic performance bound $(2m^3+5m^2+5m+2+\sqrt{9m^4+34m^3+41m^2+20m+4})/(2m(m+1)^2)$. For the container packing problem, we present an algorithm with asymptotic performance bound $2m^4+6m^3+9m^2+7m+2+\sqrt{16m^6+76m^5+141m^4+142m^3+85m^2+28m+4}/2m^2(m+1)^2$.

Not as many results have been obtained for on-line problems. Johnson *et al.* [7, 9] proved that for the on-line one-dimensional bin packing problem the asymptotic performance bound of the First Fit (FF) and the Best Fit algorithm is $(m+1)/m$, $m \geq 2$.

In this paper we present approximation algorithms for the following on-line parametric packing problems. For the *strip packing problem* we present an algorithm with asymptotic performance bound close to $(m+2)/(m+1) + 1/(m+1)^2$. For the *two-dimensional bin packing problem*, and the *three-dimensional packing problem* we present algorithms with asymptotic performance bound close to $((m+2)/(m+1))^2 + 2/(m(m+1))$. For the *container packing problem* we describe an algorithm with asymptotic performance bound close to $(m^4+5m^3+10m^2+7m+2)/(m^2(m+1)^2)$. To our knowledge, the bounds presented here are the best known for on-line parametric packing problems ($m \geq 2$). These results improve

the results of Coppersmith and Raghavan [3] for the on-line parametric two-dimensional bin packing and the on-line parametric container packing problem.

In the next section we present some definitions and notation, and in the subsequent sections we present the approximation algorithms we have developed.

2 Definitions and Notation

In this paper we consider the Euclidean space $\mathbb{R}^3$ with the xyz coordinate system to represent the packings. An item (or recipient) e has its dimensions defined by $x(e)$, $y(e)$ and $z(e)$, where each of these dimensions is the measure in the corresponding axis of the xyz coordinate system. For the two-dimensional bin packing problem, one of these coordinate axis is not considered, and for the one-dimensional bin packing problem only one of these coordinates is considered.

The *area* of a rectangle $r_i = (x_i, y_i)$ is denoted by $S(r_i)$ (*i.e.*, $S(r_i) := x_i \cdot y_i$). The *volume* of a box $b_i = (x_i, y_i, z_i)$ is denoted by $V(b_i)$.

Given a packing $\mathcal{P}$, we denote by $H(\mathcal{P})$ the *height* of the packing $\mathcal{P}$, and by $\#(\mathcal{P})$ the *number of bins* used by the packing $\mathcal{P}$.

When considering the strip packing problem (three-dimensional packing problem) we consider that there is no rectangle (box) with height greater than a constant Z.

Given a function $f : C \to \mathbb{R}$ and a subset $C' \subseteq C$, we denote by $f(C')$ the sum $\sum_{e \in C'} f(e)$.

The notation below is convenient to represent subsets of an input list.

$$\mathcal{X}[p,q] := \{e : \ p < x(e) \leq q\}, \qquad \mathcal{Y}[p,q] := \{e : \ p < y(e) \leq q\}, \qquad \mathcal{Z}[p,q] := \{e : \ p < z(e) \leq q\},$$

$$\begin{aligned} \mathcal{C}\,[p_1, q_1 \ ; \ p_2, q_2] &:= \mathcal{X}[p_1, q_1] \cap \mathcal{Y}[p_2, q_2], \\ \mathcal{C}\,[p_1, q_1 \ ; \ p_2, q_2 \ ; \ p_3, q_3] &:= \mathcal{X}[p_1, q_1] \cap \mathcal{Y}[p_2, q_2] \cap \mathcal{Z}[p_3, q_3], \end{aligned}$$

$$\mathcal{C}_m := \mathcal{C}\left[0, \frac{1}{m} \ ; \ 0, \frac{1}{m}\right], \quad \mathcal{C}_y^x[p,q] := \{e : \ x(e) \geq y(e)\}, \quad \mathcal{C}_x^y[p,q] := \{e : \ x(e) < y(e)\}.$$

We call the value s in inequalities of the form $\#(\mathcal{P}) \leq \frac{1}{s} \cdot S(L) + C$, an *area guarantee* of the packing $\mathcal{P}$ and the value v in inequalities of the form $\#(\mathcal{P}) \leq \frac{1}{v} \cdot V(L) + C$ a *volume guarantee* of the packing $\mathcal{P}$.

2.1 One-dimensional Bin Packing Problem

In the description of an algorithm we may use some known on-line algorithms. Two of these algorithms are the NF (Next Fit) and the FF (First Fit), with asymptotic performance bound 2 and 1.7, respectively [7, 6].

Moreover, when using an on-line algorithm as a subroutine, for simplicity, we may describe the new algorithm as being off-line, assuming implicitly that a transformation into an on-line algorithm can be done. The next example for the one-dimensional bin packing problem ilustrates one such transformation.

The One-dimensional Bin Packing Problem (1BP) can be formulated as follows: given a list of items $L = (e_1, \ldots, e_n)$, each with length $x(e_i)$, and bins B with length 1, find a packing of the items of L into bins B that uses the smallest possible number of bins B.

Let us first describe the algorithm NF. Given a list of items $L = (p_1, \ldots, p_n)$, the algorithm considers each item in the order given by L and verifies whether it can be packed into the current bin B_i (the first one being B_1). While this is possible, the next items are packed into B_i. When an item cannot be packed into B_i, this item is packed into a new bin B_{i+1} that becomes the current bin. The algorithm halts when all items have been packed, returning the packing $(B_1, \ldots, B_m)$, where B_m is the last generated bin.

Now consider the algorithm INF (Improved Next Fit), also for (1BP), that uses the algorithm NF as a subroutine. This algorithm has as an input list items with dimensions not greater than $\frac{1}{m}$.

Algorithm INF

Input: A list L of (one-dimensional) items.
Output: A packing of L into bins B.

1 Partition L into sublists L_1 and L_2 in the following manner:

$$L_i \leftarrow L \cap \mathcal{T}_i, \quad i = 1, 2, 3,$$

where $\mathcal{T}_1 = \mathcal{X}[\frac{1}{m+1}, \frac{1}{m}]$, $\mathcal{T}_2 = \mathcal{X}[\frac{1}{m+2}, \frac{1}{m+1}]$ and $\mathcal{T}_3 = \mathcal{X}[0, \frac{1}{m+2}]$.

2 $\mathcal{P} \leftarrow \mathrm{NF}(L_1) \| \mathrm{NF}(L_2) \| \mathrm{NF}(L_3)$.

3 Return $\mathcal{P}$.

End Algorithm.

The algorithm INF is off-line, but can be easily transformed into an on-line algorithm, as follows.

Algorithm INF′

Input: A list $L = (e_1, \ldots, e_n)$ of items.
Output: A packing of L into bins B.

1 $\mathcal{P}_i \leftarrow \emptyset$, $i = 1, 2, 3$.

2 for $k \leftarrow 1$ to n do

2.1 Let i be an integer such that $e_k \in \mathcal{T}_i$, $i \in \{1, 2, 3\}$.

2.2 Pack e_k into a bin in $\mathcal{P}_i$ using the algorithm NF. If necessary, packs e_k into a new bin in $\mathcal{P}_i$.

3 Return $\mathcal{P}_1 \| \mathcal{P}_2 \| \mathcal{P}_3$.

End algorithm.

As mentioned previously, Several algorithms in this paper are presented as off-line algorithms but they can be transformed, as above, into an on-line algorithm. Thus, we assume that a corresponding on-line version is known.

The next theorem gives an asymptotic performance bound of the algorithm INF.

Theorem 2.1 *Let L be a list of one-dimensional items with dimensions not greater than $\frac{1}{m}$. Then*

$$\mathrm{INF}(L) \leq \alpha_m \cdot \mathrm{OPT}(L) + 3,$$

where $\alpha_m = \frac{m+2}{m+1} + \frac{1}{(m+1)^2}$.

Note that for $m = 1$ the asymptotic performance bound of the algorithm INF is 1.75, that is better than the bound 2 of the algorithm NF and a little worser than the bound 1.7 proved for the algorithm FF.

The algorithm NF, always tries to pack in the last bin (current bin), and once it is not the current bin anymore, the algorithm never visits it again. The algorithm FF can be seen as an improvement of the algorithm NF, as it tries to pack an item in the previously generated bins. It can be described as follows.

Let $\{B_1, B_2, \ldots, B_k\}$ be the packing generated by the algorithm FF, where p_{i-1} is the last item thas has been packed. To pack the item p_i, the algorithm finds the smallest index j, such that p_i can be packed into the bin B_j. If there is no such bin, the item p_i is packed into a new bin B_{k+1}. This process is repeated until all the items of L have been packed.

3 Strip Packing Problem

In this section we present two algorithms for the on-line Strip Packing Problem. We use a very common technique to develop on-line algorithms for strip packing problems: for a given p, $0 < p < 1$, we use a set $\mathcal{S} := \{p^i : i \in \mathbb{Z}, \quad i \geq 0\}$ to round up the height of the rectangles to the nearest value in $\mathcal{S}$. After this rounding, we pack each rectangle of height p^i (called i-rectangle) into levels of height p^i (called i-levels). Therefore, considering only these levels, we have a one-dimensional on-line packing problem. The set $\mathcal{S}$ is called a *rounding set*.

Now, we present the description of the first algorithm, which we call $\mathrm{OSP}_{m,p}(L)$.

Algorithm $\mathrm{OSP}_{m,p}(L)$.

Input: A list of rectangles $L \subset \mathcal{X}[0, \frac{1}{m}]$ for the SPP.

Output: A packing $\mathcal{P}$ of L into $R = (a, \infty)$.

1 Pack the rectangles $r \in L$ in the order given by L.

- **1.1** Let i be an integer such that $p^{i+1} < \frac{y(r)}{Z} \leq p^i$.
- **1.2** If $\frac{1}{m+1} < x(r) \leq \frac{1}{m}$, then pack r in the i-levels generated for these rectangles. Use the algorithm NF to pack the rectangle r in one of these levels. If necessary, pack r into a new level of height p^i.
- **1.3** If $0 < x(r) \leq \frac{1}{m+1}$, then pack r analogously as in step 1.2, but now packing r in levels generated for rectangles with $0 < x(r) \leq \frac{1}{m+1}$.

2 Return the generated packing.

End Algorithm.

Theorem 3.1 *Let $L \subset \mathcal{X}[0, \frac{1}{m}]$ be an input list for the strip packing problem, where the rectangles of L have height at most Z. Then, for any value p, $0 < p < 1$, we have*

$$\mathrm{OSP}_{m,p}(L) \leq \frac{1}{p}\left(\frac{m+1}{m}\right)\mathrm{OPT}(L) + \frac{2Z}{1-p}.$$

The algorithm $\mathrm{OSP}_{m,p}$ can be refined to have a better asymptotic performance bound. For this, we divide the list L into two sublists L_1 and L_2, and apply the algorithm $\mathrm{OSP}_{m,p}$ for the sublist L_1 and the algorithm $\mathrm{OSP}_{m+1,p}$ for the sublist L_2. Then, we return the packing that is the concatenation of these two packings. We denote this algorithm as $\mathrm{IOSP}_{m,p}$ (Improved OSP).

Theorem 3.2 *Let L be an input list for the strip packing problem, containing rectangles with length at most $\frac{1}{m}$ and height at most Z. Then, for any given p, $0 < p < 1$, we have*

$$\mathrm{IOSP}_{m,p}(L) \leq \alpha_m \cdot \mathrm{OPT}(L) + \frac{4Z}{1-p},$$

where $\lim_{p\to 1} \alpha_m = \frac{m+2}{m+1} + \frac{1}{(m+1)^2}$.

4 Two Dimensional Bin Packing Problem

In this section we present on-line algorithms to pack small rectangles into two-dimensional bins. The on-line algorithm with the best known asymptotic performance bound for this problem is due to Li and Cheng [10] and Csirik and van Vliet [5] with bound 2.86.

First, we describe a family of algorithms $\mathrm{FF}_p^{(2)}$, with parameter $0 < p < 1$, used as subroutines, that use a rounding set $\mathcal{S} = \{1, p, p^2, \ldots, p^i, \ldots\}$. In these algorithms, the width of each rectangle is rounded up to the next value in $\mathcal{S}$, say $p^i \in \mathcal{S}$. Furthermore, each rectangle is packed as a one-dimensional item. We describe below this algorithm.

Algorithm $\mathrm{FF}_p^{(2)}$
Input: A list of rectangles $L = (r_1, \ldots, r_n)$.
Output: A packing of L into bins $R = (1, 1)$.

1 For $j \leftarrow 1$ to n do

1.1 Let i be an integer such that r_j is an i-rectangle.

1.2 Consider the rectangle r_j as a bin with width $x(r_j)$ and consider as bins of length 1, the i-levels generated until this step. Use the algorithm FF to pack r_j into the i-levels.

1.3 If a new i-level is generated in step 1.2, say $p^i \in \mathcal{S}$, then use the algorithm FF to pack this i-level inside the two dimensional bins. Otherwise, pack the level into a new bin.

2 Return the packing generated in step 1.

End Algorithm.

This algorithm packs the rectangles side by side along the x-axis. We also denote this algorithm as $\mathrm{FF}_p^{(x2)}$. The analogous version of this algorithm $\mathrm{FF}_p^{(2)}$ that generates a packing with strips in the y direction is denoted by $\mathrm{FF}_p^{(y2)}$.

Li and Cheng proved that the algorithm $\mathrm{FF}_p^{(2)}$ has an asymptotic performance bound that can be made as close to 2.89 as desired, but for that one has to take p close to 1 and such that $p^s = \frac{1}{2}$ for some s.

Now, we describe an on-line algorithm to pack small rectangles. The basic idea is to guarantee a minimum area utilization in each bin, for rectangles of relatively big size (using only list subdivision). For smaller rectangles we use the algorithm $\mathrm{FF}_p^{(2)}$.

Algorithm $\mathrm{O2D}_m$

Input: A list of rectangles $L = (r_1, \ldots, r_n)$ with $x(r_i) \leq \frac{1}{m}$ and $y(r_i) \leq \frac{1}{m}$.

Output: A packing of L into bins $R = (1, 1)$.

1 Subdivide the list L into sublists $L_1, \ldots, L_6$ in the following way:

$$\begin{aligned}
L_1 &\leftarrow L \cap \mathcal{C}\left[\tfrac{1}{m+1}, \tfrac{1}{m} \;;\; \tfrac{1}{m+1}, \tfrac{1}{m}\right], \\
L_2 &\leftarrow L \cap \mathcal{C}\left[0, \tfrac{1}{m+1} \;;\; \tfrac{1}{m+1}, \tfrac{1}{m}\right], \\
L_3 &\leftarrow L \cap \mathcal{C}\left[\tfrac{1}{m+1}, \tfrac{1}{m} \;;\; 0, \tfrac{1}{m+1}\right], \\
L_4 &\leftarrow L \cap \mathcal{C}\left[0, \tfrac{1}{m+1} \;;\; \tfrac{1}{m+2}, \tfrac{1}{m+1}\right] \cap \mathcal{C}_x^y, \\
L_5 &\leftarrow L \cap \mathcal{C}\left[\tfrac{1}{m+2}, \tfrac{1}{m+1} \;;\; 0, \tfrac{1}{m+1}\right] \cap \mathcal{C}_y^x, \\
L_6 &\leftarrow L \cap \mathcal{C}\left[0, \tfrac{1}{m+2} \;;\; 0, \tfrac{1}{m+2}\right].
\end{aligned}$$

2 $\mathcal{P}_i \leftarrow \mathrm{FF}_{1/m}^{(x2)}(L_i), \quad i = 1, 2.$

3 $\mathcal{P}_3 \leftarrow \mathrm{FF}_{1/m}^{(y2)}(L_3).$

4 $\mathcal{P}_4 \leftarrow \mathrm{FF}_{1/(m+1)}^{(x2)}(L_4).$

5 $\mathcal{P}_5 \leftarrow \mathrm{FF}_{1/(m+1)}^{(y2)}(L_5).$

6 $\mathcal{P}_6 \leftarrow \mathrm{FF}_p^{(2)}(L_6), \quad$ for $\quad p = \frac{(m+2)^2 m^2}{(m+1)^2 (m+1)^2}.$

7 $\mathcal{P} \leftarrow \bigcup_{i=1}^{6} \mathcal{P}_i.$

8 Return $\mathcal{P}$.

End Algorithm.

The following theorem shows that the algorithm $\mathrm{O2D}_m$ has asymptotic performance bound $((m+1)/m)^2$. The drawback of this algorithm is the additive constant β that depends on m. When m increases, the asymptotic performance bound becomes very close to 1, but the value of β increases. Note that if we fix p as a constant—instead of a function of m—the factor we loose because of the rounding also turns out to be constant.

Lemma 4.1 *For any list L of rectangles, with dimensions at most $\frac{1}{m}$, the following holds:*

$$\mathrm{O2D}_m(L) \leq \left(\frac{m+1}{m}\right)^2 \cdot S(L) + \mathcal{O}(m^2).$$

Using this lemma, it is easy to obtain the following result.

Theorem 4.2 *For any list L of rectangles, with dimensions at most $\frac{1}{m}$, the following holds:*

$$\mathrm{O2D}_m(L) \leq \left(\frac{m+1}{m}\right)^2 \cdot \mathrm{OPT}(L) + \mathcal{O}(m^2).$$

This algorithm can also be refined yielding an algorithm with a better asymptotic performance bound. This can be done using the same approach we used for the algorithm $\mathrm{IOSP}_{m,p}$ with respect to the algorithm $\mathrm{OSP}_{m,p}$. We call it algorithm $\mathrm{IO2D}_m$. In this algorithm the list L is subdivided into sublists $L_1, \ldots, L_6$, as indicated in in Figure 1.

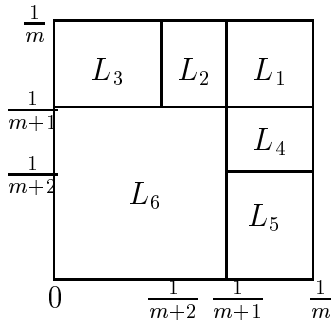

Figure 1: Partition of the list L performed by the algorithm $\mathrm{IO2D}_m$.

Then, the following strategy is used. For the list L_1 the algorithm packs m^2 rectangles in each bin. In this case we have an asymptotic optimum packing for the list L_1 with area guarantee $(m/m+1)^2$. For the lists $L_2, \ldots, L_5$, the algorithm applies a variant of the algorithm $\mathrm{FF}_p^{(2)}$ for each sublist (as done for algorithm O2D) obtaining an area guarantee of at least $m \cdot (m+1) \cdot (1/(m+1)) \cdot (1/(m+2)) = m/(m+2)$. For each sublist L_6, the algorithm applies the algorithm $\mathrm{O2D}_{m+1}$, obtaining for this packing an area guarantee of at least $((m+2)/(m+1))^2 \geq m/(m+2)$. At last, the algorithm returns a concatenation of the packings generated for each sublist. Note that we have a packing of L divided into two parts. For one part we have an asymptotic optimum packing with area guarantee $(m/(m+1))^2$, and for the other part we have a packing with area guarantee $m/(m+2)$. Therefore, the asymptotic performance bound of the algorithm $\mathrm{IO2D}_m$ is $\alpha_m = \{h_1 + h_2\}/\{\max\{h_1, \left(\frac{m}{m+1}\right)^2 h_1 + \left(\frac{m}{m+2}\right) h_2\}$. Thus, we have the following result.

Theorem 4.3 *For any list L of rectangles with dimensions at most $\frac{1}{m}$,*

$$\mathrm{IO2D}_m(L) \leq \alpha_m \cdot \mathrm{OPT}(L) + \mathcal{O}(m^2),$$

where $\alpha_m \leq \left(\frac{m+2}{m+1}\right)^2 + \frac{2}{m(m+1)}$.

5 Three Dimensional Packing Problem

Li and Cheng [11] were the first authors to develop approximation algorithms for this problem. In [13] they present the algorithm with the best known on-line bound of 2.89. In this section

we present an algorithm for packing boxes with bottom dimensions not greater than $\frac{1}{m}$, whose asymptotic performance bound is $(m^4 + 5m^3 + 10m^2 + 7m + 2)/(m^2(m+1)^2)$.

First, we present the algorithm $\mathrm{OTP}_{m,p}$ and then its improved version, called $\mathrm{IOTP}_{m,p}$.

These algorithms use a rounding procedure analogous to the one used by the algorithm $\mathrm{OSP}_{m,p}$ and $\mathrm{IOSP}_{m,p}$. That is, the height of each box is rounded up to the nearest value in a set $\mathcal{S} = \{1, p, p^2, \ldots, p^i, \ldots\}$. Each box of height p^i, called i-box, is first packed in a level of height p^i, called i-level.

Algorithm $\mathrm{OTP}_{m,p}$

Input: A list of boxes $L = (b_1, \ldots, b_n)$, $b_i \in \mathcal{C}\left[0, \frac{1}{m}\ ;\ 0, \frac{1}{m}\right]$.

Output: A packing of boxes in L into a box $B = (1, 1, \infty)$.

1 $\mathcal{P} \leftarrow \emptyset$.

2 For $i = 1$ until n do

2.1 Let $j \geq 0$ be an integer such that b_i is a j-box.

2.2 Let $\mathcal{N}_j$ be the set of j-levels generated so far.

2.3 Use the algorithm $\mathrm{O2D}_m$ to pack b_i in levels $\mathcal{N}_j$, visualizing each level of $\mathcal{N}_j$ as a rectangle (bin) $(1, 1)$ and b_i as a rectangle $(x(b_i), y(b_i))$. If necessary, generate a new j-level into $\mathcal{P}$ (or in a new bin) to pack b_i.

3 Return $\mathcal{P}$.

End algorithm.

The following result can be proved about the algorithm $\mathrm{OTP}_{m,p}$.

Theorem 5.1 *Let $m \geq 2$ be an integer, and p a positive number less than 1. Let $L \subset \mathcal{C}_m$ be the list of boxes to be packed into a box $B = (1, 1, \infty)$. Then, the algorithm $\mathrm{OTP}_{m,p}$ for the three-dimensional packing problem is such that*

$$\mathrm{OTP}_{m,p}(L) \leq \frac{1}{p}\left(\frac{m+1}{m}\right)^2 \cdot \mathrm{OPT}(L) + \mathcal{O}\left(\frac{m^2}{1-p}\right).$$

The improved version of this algorithm can be obtained by using an approach similar to the ones we have mentioned in the previous algorithms.

Using a rounding procedure in the height of the boxes, as done for the algorithm $\mathrm{OTP}_{m,p}$, and considering the packings generated in the same way done to algorithm $\mathrm{IO2D}_m$, we can build an on-line algorithm with an asymptotic performance bound as close to the bound of the algorithm $\mathrm{IO2D}_m$ as desired. The strategy is to pack the boxes of height p^i in levels of height p^i. The packing in levels is done by algorithm $\mathrm{IO2D}_m$ (for 2BP). Now the final packing is constituted by two parts. An asymptotically optimum packing with area guarantee $\frac{1}{p}\left(\frac{m}{m+1}\right)^2$ and another one with area guarantee $\frac{1}{p}\left(\frac{m}{m+2}\right)$. We call this algorithm $\mathrm{IOTP}_{m,p}$. Therefore, the following result is valid.

Theorem 5.2 *Let L be a list of boxes with bottom dimensions at most $\frac{1}{m}$. Then, for any given p, $0 < p < 1$, the following holds:*

$$\mathrm{IOTP}_{m,p}(L) \leq \alpha_{m,p} \cdot \mathrm{OPT}(L) + \mathcal{O}\left(\frac{m^2}{1-p}\right),$$

where $\lim_{p\to 1} \alpha_{m,p} = \alpha(\mathrm{IOTP}_{m,p}) \leq \left(\frac{m+2}{m+1}\right)^2 + \frac{2}{m(m+1)}$.

6 Container Packing Problem

Coppersmith and Raghavan [3] were the first authors to present an approximation algorithm for teh container packing problem. They presented an algorithm with asymptotic performance bound 6.25. The algorithm with the best known asymptotic performance bound for this problem is due to Li and Cheng [10] and Csirik and van Vliet [5], with asymptotic performance bound 4.84.

We describe an algorithm for the 3BP, called H3D (Hibrid 3-D bin packing). This algorithm uses the same strategy used by the algorithm HFF (Hibrid first fit [1]). The algorithm H3D generates the packing in the same way, and uses an algorithm for the three-dimensional packing proble (TPP) to generate levels, and then uses an algorithm for the one-dimensional bin packing problem (*upp*) to pack the levels into the containers. The algorithms for TPP and 1BP are not fixed, but given as input subroutines of the algorithm H3D.

Algorithm $\mathrm{H3D}^z$

Input: A list of boxes L.

Subroutines: Algorithm $\mathcal{A}_{\mathrm{TPP}}$ for TPP and algorithm $\mathcal{A}_{\mathrm{UNI}}$ for 1BP.

Output: A packing of L into containers $B = (a, b, c)$.

1 $\mathcal{P} \leftarrow \mathcal{A}_{\mathrm{TPP}}(L)$.

2 Let $\mathcal{N}$ be the set of levels in $\mathcal{P}$.

3 Apply algorithm $\mathcal{A}_{\mathrm{UNI}}$ to pack the levels of $\mathcal{N}$ into containers B. Each level $N \in \mathcal{N}$, with height z_N, is seen as a one-dimensional item of height z_N, and each container B is seen as a one-dimensional bin of height 1. Let $\mathcal{P}_{\mathrm{H3D}}$ be the packing generated with this procedure.

4 Return $\mathcal{P}_{\mathrm{H3D}}$.

End algorithm.

Note that if the algorithm $\mathcal{A}_{\mathrm{UNI}}$ and $\mathcal{A}_{\mathrm{TPP}}$ are on-line, then the algorithm $\mathrm{H3D}^z$ can also be transformed into an on-line algorithm.

Denote by $\mathrm{H3D}^x$ and $\mathrm{H3D}^y$ the variants of this algorithm where the level generation and the one-dimensional packing is done in the directions x and y, respectively.

The next algorithm uses an analogous idea behind the algorithm $\mathrm{O2D}_m$. The input list L is partitioned into three sublists, and the algorithm maintains three types of containers, one for each sublist. The boxes of the first list have height in the interval $(1/(m+1), 1/m]$. Therefore, in each container m levels are allocated. The boxes from the first list are packed into levels of this type using the algorithm $\mathrm{O2D}_m$. The same procedure is used for the second list, with boxes having height in the interval $(1/(m+2), 1/(m+1)]$. For the third list, it is applied a

rounding procedure using a power of p, $0 < p < 1$, as done in the algorithm O2D$_m$. Again, in order not to loose a constant factor because of the roundings, p is taken as a function of m.

Algorithm O3D$_m$
Input: A list of boxes L with dimensions at most $\frac{1}{m}$.
Output: A packing of L into containers $B = (1, 1, 1)$.

1 Subdivide the list L into sublists $L_1, \ldots, L_3$ in the following way

$L_1 \leftarrow L \cap \mathcal{Z}[\frac{1}{m+1}, \frac{1}{m}]$;
$L_2 \leftarrow L \cap \mathcal{Z}[\frac{1}{m+2}, \frac{1}{m+1}]$;
$L_3 \leftarrow L \setminus (L_1 \cup L_2)$.

2 Use the following procedure to generate a packing $\mathcal{P}_1$ of the sublist L_1. Consider the levels of height $\frac{1}{m}$ generated so far. Pack the next box of L_1 in these levels using the algorithm O2D$_m$. If necessary, generate a new level with height $\frac{1}{m}$ in the last container, or in a new container (if needed).

3 Generate a packing $\mathcal{P}_2$ of the sublist L_2 using a procedure analogous to the one used in step 2, but now with levels of height $\frac{1}{m+1}$.

4 $\mathcal{P}_3 \leftarrow$ H3D^z(OTP$_{m,p}, L_3,$ FF), with $p = \frac{m(m+2)}{(m+1)^2}$;

5 $\mathcal{P} \leftarrow \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_3$.

6 Return $\mathcal{P}$.

End algorithm.

Again, we note that this algorithm is not on-line, but its transformation into an on-line algorithm is simple.

Lemma 6.1 *For any list L of boxes with dimensions at most $\frac{1}{m}$, $m \geq 2$, we have*

$$\text{O3D}_m(L) \leq \left(\frac{m+1}{m}\right)^3 V(L) + \mathcal{O}(m^4).$$

Using the above lemma, the following result is immediate.

Theorem 6.2 *For any list L of boxes with dimensions at most $\frac{1}{m}$, $m \geq 2$, the following holds:*

$$\text{O3D}_m(L) \leq \left(\frac{m+1}{m}\right)^3 \text{OPT}(L) + \mathcal{O}(m^4).$$

Using the same strategy used in the algorithm IO2D$_m$, we can refine the algorithm O3D$_m$ obtaining another one with a better asymptotic performance bound. Let us call IO3D$_m$ this improved algorithm. This algorithm first divides the list L into two sublists, L_1 and L_2, where

$$\begin{aligned} L_1 &\leftarrow L \cap \mathcal{X}[\frac{1}{m+1}, \frac{1}{m}] \cap \mathcal{Y}[\frac{1}{m+1}, \frac{1}{m}] \cap \mathcal{Z}[\frac{1}{m+1}, \frac{1}{m}], \\ L_2 &\leftarrow L \setminus L_1. \end{aligned}$$

For the list L_1, this algorithm packs m^3 boxes into each container, except perhaps in the last. As each box has volume greater than $(1/(m+1))^3$, the packing of L_1 has volume guarantee of at least $(m/(m+1))^3$.

Subdividing the list L_2 and using the algorithms IO2D$_m$ and O3D$_m$ we can obtain a packing of L_2 with a volume guarantee of at least $m^2/((m+1)(m+2))$. At last, the algorithm IO3D$_m$ returns the concatenation of the packings of the sublists L_1 and L_2.

The following result can be proved for this algorithm.

Theorem 6.3 *Let L be any list of boxes with dimensions at most $\frac{1}{m}$, $m \geq 2$, Then the following holds:*

$$\text{IO3D}_m(L) \leq \alpha_m \cdot \text{OPT}(L) + \mathcal{O}(m^4),$$

where $\alpha_m \leq (m^4 + 5m^3 + 10m^2 + 7m + 2)/(m^2(m+1)^2)$.

7 Conclusion

The asymptotic performance bounds of the algorithms we presented in this paper are summarized in the following table. We note that the bounds are the best known for on-line algorithms for each problem considered here.

Problem	Best Known	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$
Strip Packing.	1.691 [5]	1.750	1.445	1.313	1.24	1.195	1.164	1.141
2D bin packing	2.86 [1]	3.25	2.112	1.73	1.54	1.428	1.354	1.302
3D Packing	2.89 [15]	3.25	2.112	1.73	1.54	1.428	1.354	1.302
Container packing	4.84 [5, 10]	6.25	3.112	2.285	1.915	1.708	1.576	1.486

Table 1: Asymptotic performance bounds of the on-line algorithms mentioned.

References

[1] 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.

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

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

[4] J. Csirik. The parametric behavior of the first-fit decreasing bin packing algorithm. *J. of Algorithms*, 15:1–28, 1993.

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

[6] M. R. Garey, R. L. Graham, and J. D. Ullman. Worst-case analysis of memory allocation algorithms. In *Proc. 4th Annual ACM Symp. on the Theory of Computing*, pages 143–150, 1972.

[7] D. S. Johnson. *Near-optimal bin packing algorithms.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Mass., 1973.

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

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

[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. On three-dimensional packing. *SIAM J. on Comput.*, 19:847–867, 1990.

[12] K. Li and K-H. Cheng. Static job scheduling in partitionable mesh connected systems. *J. Parallel and Distributed Computing*, 10:152–159, 1990.

[13] K. Li and K-H. Cheng. Heuristic algorithms for on-line packing in three dimensions. *J. of Algorithms*, 13:589–605, 1992.

[14] F. K. Miyazawa and Y. Wakabayashi. Approximation algorithms for the orthogonal z-oriented 3-D packing problem. To appear in *SIAM J. on Computing.*

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

[16] F. K. Miyazawa and Y. Wakabayashi. Approximation algorithms for packing small items. In *XX Congresso Nacional de Matemática Aplicada e Computacional (Mini Simpósio de Corte e Empacotamento)*, pages 7–144, 1997.