A New Upper Bound 2.5545 on 2D Online Bin Packing

XIN HAN, Dalian University of Technology
FRANCIS Y. L. CHIN and HING-FUNG TING, The University of Hong Kong
GUOCHUAN ZHANG, Zhejiang University
YONG ZHANG, The University of Hong Kong

The 2D Online Bin Packing is a fundamental problem in Computer Science and the determination of its asymptotic competitive ratio has research attention. In a long series of papers, the lower bound of this ratio has been improved from 1.808, 1.856 to 1.907 and its upper bound reduced from 3.25, 3.0625, 2.8596, 2.7834 to 2.66013. In this article, we rewrite the upper bound record to 2.5545. Our idea for the improvement is as follows.

In 2002, Seiden and van Stee [Seiden and van Stee 2003] proposed an elegant algorithm called $H \otimes C$, comprised of the *Harmonic algorithm* H and the *Improved Harmonic algorithm* C, for the two-dimensional online bin packing problem and proved that the algorithm has an asymptotic competitive ratio of at most 2.66013. Since the best known online algorithm for one-dimensional bin packing is the *Super Harmonic algorithm* [Seiden 2002], a natural question to ask is: could a better upper bound be achieved by using the Super Harmonic algorithm instead of the Improved Harmonic algorithm? However, as mentioned in Seiden and van Stee [2003], the previous analysis framework does not work. In this article, we give a positive answer for this question. A new upper bound of 2.5545 is obtained for 2-dimensional online bin packing. The main idea is to develop new weighting functions for the Super Harmonic algorithm and propose new techniques to bound the total weight in a rectangular bin.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*2D bin packing*

General Terms: Algorithms

Additional Key Words and Phrases: Online algorithms, bin packing problems, competitive ratio

ACM Reference Format:
Han, X., Chin, F. Y. L., Ting, H.-F., Zhang, G., and Zhang, Y. 2011. A new upper bound 2.5545 on 2D online bin packing. ACM Trans. Algor. 7, 4, Article 50 (September 2011), 18 pages.
DOI = 10.1145/2000807.2000818 http://doi.acm.org/10.1145/2000807.2000818

1. INTRODUCTION

In two-dimensional bin packing, each item (w_i, h_i) is a rectangle of width $w_i \leq 1$ and height $h_i \leq 1$. Given a list of such rectangular items, one is asked to pack all of them into a minimum number of square bins of side length one so that their sides are parallel to the sides of the bin. Rotation is not allowed. The problem is clearly strongly

X. Han was partially supported by the Fundamental Research Funds for the Central Universities and NFSC (11101065). G. Zhang was partially supported by NSFC (10971192).
Authors' addresses: X. Han, Software School, Dalian University of Technology, Road 8, Economy and Technology Development Zone, Dalian, P.R. China, 116620; email: hanxin.mail@gmail.com; F. Y. L. Chin, H.-F. Ting, and Y. Zhang, Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong, China; email: {chin, hfting, yzhang}@cs.hku.hk; G. Zhang, College of Computer Science, Zhejiang University, ZheDa Road 38, Hangzhou 310027, China; email: zgc@zju.edu.cn.

DOI 10.1145/2000807.2000818 http://doi.acm.org/10.1145/2000807.2000818

NP-hard since it is a generalization of the one-dimensional bin packing problem [Coffman et al. 1987]. In this article, we will consider the online version of two-dimensional bin packing, in which the items are released one by one and we must irrevocably pack the current item into a bin without any information on the next items. Before presenting the previous results and our work, we first review the standard measure for online bin packing algorithms.

Asymptotic Competitive Ratio. To evaluate an online algorithm for bin packing problems, we use the *asymptotic competitive ratio* defined as follows. Consider an online algorithm A. For any list L of items, let $A(L)$ be the cost (number of bins used) incurred by algorithm A and let $OPT(L)$ be the corresponding optimal value. Then, the *asymptotic competitive ratio* for algorithm A is

$$R_A^\infty = \lim_{k\to\infty} \sup \max_L \{A(L)/OPT(L) | OPT(L) = k\}.$$

Previous Work. Bin packing has been well-studied. For the one-dimensional case, Johnson et al. [1974] showed that the First Fit algorithm (FF) has an asymptotic competitive ratio of 1.7. Yao [1980] improved algorithm FF with a better upper bound of 5/3. Lee and Lee [1985] introduced the class of Harmonic algorithms, for which an asymptotic competitive ratio of 1.63597 was achieved. Ramanan et al. [1989] further improved the upper bound to 1.61217. The best known upper bound so far is from the Super Harmonic algorithm by Seiden [2002] whose asymptotic competitive ratio is at most 1.58889. As for the negative results, Yao [1980] showed that no online algorithm has asymptotic competitive ratio less than 1.5. Brown [1979] and Liang [1980] independently provided a better lower bound of 1.53635. The best known lower bound to date is 1.54014 [van Vliet 1992].

As for two-dimensional online bin packing, a lower bound of 1.6 was given by Galambos [1991]. The result was gradually improved to 1.808 [Galambos and van Vliet 1994], 1.857 [van Vliet 1995], and 1.907 [Blitz et al. 1996]. Coppersmith and Raghan [1989] gave the first online algorithm with asymptotic competitive ratio 3.25. Csirik et al. [1993] improved the upper bound to 3.0625. Csirik and van Vliet [1993] presented an algorithm for all d dimensions, where in particular for two dimensions, they obtained a ratio of at most 2.8596. Based on the techniques of the Improved Harmonic, Han et al. [2001] improved the upper bound to 2.7834. The best known online algorithm to date is the one called $A \otimes C$ presented by Seiden and van Stee [2003], where A and C stand for two one-dimensional online bin packing algorithms. Basically, A and C are applied to one dimension of the items with rounding sizes. In this seminal paper, Seiden and van Stee proved that the asymptotic competitive ratio of $H \otimes C$ is at most 2.66013, where H is the Harmonic algorithm [Lee and Lee 1985] and C is an instance of the improved Harmonic algorithm. It has been open since then to improve the upper bound. A natural idea is to use an instance of the Super Harmonic algorithm [Seiden 2002] instead of the improved Harmonic algorithm. However, as mentioned in Seiden and van Stee [2003], in that case, the previous analysis framework cannot be extended to Super Harmonic.

We also briefly overview the offline results on two-dimensional bin packing. Chung et al. [1982] showed an approximation algorithm with an asymptotic performance ratio of 2.125. Caprara [2002] improved the upper bound to 1.69103. Bansal et al. [2009] derived a randomized algorithm with asymptotic performance ratio of at most 1.525. As for the negative results, Bansal et al. [2006] showed that the two-dimensional bin packing problem does not admit an asymptotic polynomial-time approximation scheme.

For the special case where items are squares, there is also a large number of results [Coppersmith and Paghavan 1989; Seiden and van Stee 2003; Miyazawa and Wakabayashi 2003; Epstein and van Stee 2004, 2005b, 2005a; Han et al. 2006].

Especially for bounded space online algorithms, Epstein and van Stee [2005b] gave an optimal online algorithm.

Our Contributions. There are two main contributions in this article.

—We revisit the 1D online bin-packing algorithm: Super Harmonic, and give new weighting functions for it, which are much simpler than the ones introduced in Seiden [2002], and the new weighting functions have interests in its own.
—We generalize the previous analysis framework for 2D online bin-packing algorithms used in Seiden and van Stee [2003], and show that the new analysis framework is very useful in analyzing 2D or multidimensional online bin-packing problems.

By combining the new weighting functions with the new analysis framework, we design a new 2D online bin-packing algorithm with a competitive ratio 2.5545, which improves the previous bound of 2.66013 in 2002 [Seiden and van Stee 2003]. As mentioned in Seiden and van Stee [2003], the old analysis framework does not work well with the old weighting functions in Seiden [2002], that is, the old approach does not guarantee an upper bound better than 2.66013. This is testified in the following way: consider our algorithm, if we use old weighting functions with the old framework to analyze it, the competitive ratio is at least 3.04, and if we use the old weighting functions with the new framework, the competitive ratio is at least 2.79.

Organization of This Article. Section 2 will review the Harmonic and Super Harmonic algorithms as preliminaries. Section 3 defines the weighting functions for Super Harmonic. Section 4 describes and analyzes the two-dimensional online bin-packing algorithm $H \otimes SH+$. Section 5 concludes.

2. PRELIMINARIES

We first review two online algorithms for one-dimensional bin packing, Harmonic and Super Harmonic, which are employed in designing online algorithms for two-dimensional bin packing.

2.1. The Harmonic Algorithm

The Harmonic algorithm is a fundamental bin packing algorithm with a simple and nice structure, that was introduced by Lee and Lee [1985]. The algorithm works as follows. Given a positive integer k, each item is immediately classified into one of k types according to its size upon its arrival. In particular, if an item has a size in interval $(1/(i+1), \frac{1}{i}]$ for some integer i, where $1 \le i < k$, then it is a type i item; otherwise, it is of type k. The type i item is then packed, using the simple Next Fit (NF) algorithm, into the open (not fully packed) bin designated to contain type i items exclusively; new bins are opened when necessary. At any time, there is at most one open bin for each type and any closed (fully packed) bin for type i is packed exactly with i items of type i for $1 \le i < k$.

For an item of size x, we define a weighting function $W_H(x)$ for the Harmonic algorithm as follows:

$$W_H(x) = \begin{cases} \frac{1}{i}, & \text{if } \frac{1}{i+1} < x \le \frac{1}{i} \text{ with } 1 \le i < k. \\ \frac{k}{k-1}x & \text{if } 0 < x \le \frac{1}{k}. \end{cases}$$

The following lemma is directly from Lee and Lee [1985].

LEMMA 2.1. *For any list L, we have*

$$H(L) \le \sum_{p \in L} W_H(p) + O(1),$$

where $H(L)$ is the number of bins used by the Harmonic algorithm for list L.

2.2. The Super Harmonic Algorithm

The Super Harmonic algorithm [Seiden 2002] is a generalization of the Improved Harmonic algorithm and the Harmonic algorithm. Super Harmonic first classifies each item into one of $k+1$ types, where k is a positive integer, and then assigns to the item a color of either blue or red. It allows items of up to two different types to share the same bin. In any one bin, all items of the same type have same color and items of different type have different colors. For items of type i ($i \leq k$), the algorithm maintains two parameters β_i and γ_i to bound respectively the number of blue items and the number of red items in a bin. More details are given in this section.

Classification into Types. Let $t_1 = 1 > t_2 > \cdots > t_k > t_{k+1} = \epsilon > t_{k+2} = 0$ be real numbers. An interval I_i is defined to be $(t_{i+1}, t_i]$, for $i = 1, \ldots, k+1$. An item with size x is of type i if $x \in I_i$.

Coloring Red or Blue. Each type i item is also assigned a color, either red or blue, for $i \leq k$. The algorithm uses two sets of counters, $e_1, \ldots, e_k$ and $s_1, \ldots, s_k$, all of which are initially zero. The total number of type i items is denoted by s_i, while the number of type i red items is denoted by e_i. For $1 \leq i \leq k$, during the packing process, the fraction of type i items that are red is maintained, that is, $e_i = \lfloor \alpha_i s_i \rfloor$, where $\alpha_1, \ldots, \alpha_k \in [0,1]$ are constants.

Maximal Number of Blue Items. Let $\beta_i = \lfloor 1/t_i \rfloor$ for $1 \leq i \leq k$, that is the maximal number of blue items of type i that can be accepted in a single bin.

Space Left for Red Items. Let $\delta_i = 1 - t_i\beta_i$, which is the lower bound of the space left when a bin consists of β_i blue items of type i. If possible, we want to use the space left for small red items. Note that, in the algorithm, in order to simplify the analysis, instead of using δ_i, less space is used, namely $D = \{\Delta_0, \Delta_1, \ldots, \Delta_K\}$, as the spaces into which red items can be packed, where $0 = \Delta_0 < \Delta_1 < \cdots < \Delta_K < 1/2$ and $K \leq k$. Let $\Delta_{\phi(i)}$ be the space to be used to accommodate red items in a bin that holds β_i blue items of type i, where function ϕ is defined as $\{1,\ldots,k\} \mapsto \{0,\ldots,K\}$ such that $\phi(i)$ is the maximum number such that $\Delta_{\phi(i)} \leq \delta_i$. If $\phi(i) = 0$, then no red items are accepted.

We set $\alpha_i = 0$ if $t_i > \Delta_K$. For convenient use in our analysis in the next section, we introduce a function called $\varphi(i)$, which gives the index of the smallest space in D into which a red item of type i can be placed:

$$\varphi(i) = \min\{j \mid t_i \leq \Delta_j, 1 \leq j \leq K\}.$$

Maximal Number of Red Items. Now we define γ_i. Let $\gamma_i = 0$ if $t_i > \Delta_K$; otherwise, $\gamma_i = max\{1, \lfloor \Delta_1/t_i \rfloor\}$, that is, if $\Delta_1 < t_i \leq \Delta_K$, we set $\gamma_i = 1$, otherwise, $\gamma_i = \lfloor \Delta_1/t_i \rfloor$.

Naming Bins. It is also convenient to name the bins by groups as follows.

$$\{(i) \mid \phi_i = 0, 1 \leq i \leq k\}.$$

$$\{(i, ?) \mid \phi_i \neq 0, 1 \leq i \leq k\}.$$

$$\{(?, j) \mid \alpha_j \neq 0, 1 \leq j \leq k\}.$$

$$\{(i, j) \mid \phi_i \neq 0, \alpha_j \neq 0, \gamma_j t_j \leq \Delta_{\phi(i)}, 1 \leq i, j \leq k\}.$$

Group (i) consists of bins that hold only blue items of type i. Group (i, j) consists of bins that contain blue items of type i and red items of type j. Blue group $(i, ?)$ and red group $(?, j)$ are indeterminate bins *currently* containing only blue items of type i or red items of type j respectively. During packing, red items or blue items will be packed into indeterminate bins if necessary, that is, indeterminate bins will be changed into (i, j).

The Super Harmonic algorithm is outlined as follows.

Super Harmonic Algorithm

(1) For each item $p : i \leftarrow$ type of p,
  (a) if $i = k + 1$ then use NF algorithm,
  (b) else $s_i \leftarrow s_i + 1$; if $e_i < \lfloor \alpha_i s_i \rfloor$ then $e_i \leftarrow e_i + 1$; { color p red }
    i. If there is a bin in group $(?, i)$ with fewer than γ_i type i items, then place p in it. Else if, for any j, there is a bin in group (j, i) with fewer than γ_i type i items then place p in it.
    ii. Else if there is some bin in group $(j, ?)$ such that $\Delta_{\phi(j)} \geq \gamma_i t_i$, then pack p in it and change the bin into (j, i).
    iii. Otherwise, open a bin $(?, i)$, pack p in it.
  (c) else {color p blue}:
    i. if $\phi_i = 0$ then if there is a bin in group i with fewer than β_i items then pack p in it, else open a new group i bin, then pack p in it.
    ii. Else:
      A. if, for any j, there is a bin in group (i, j) or $(i, ?)$ with fewer than β_i type i items, then pack p in it.
      B. Else if there is a bin in group $(?, j)$ such that $\Delta_{\phi(i)} \geq \gamma_j t_j$ then pack p in it, and change the group of this bin into (i, j).
      C. Otherwise, open a new bin $(i, ?)$ and pack p in it.

3. NEW WEIGHTING FUNCTIONS FOR SUPER HARMONIC

In this section, we develop new weighting functions for Super Harmonic that are simpler than the weighting system in Seiden [2002]. The weighting functions will be useful in analyzing the proposed online algorithm as we shall see in the next section.

3.1. Intuitions for Defining Weights

Weighting functions are widely used in analyzing online bin-packing problems. Roughly speaking, for an item, the value by one of weight functions is the fraction of a bin occupied by the item in the online algorithm. There is a constraint in defining weights for items for an online algorithm, which will be given later. We will use $K+1$ weighting functions. Let $W^i(p)$ be the ith weight of an item p, where $1 \leq i \leq K+1$. For any input L, the constraint is

$$A(L) \leq \max_{1 \leq i \leq K+1} \left\{ \sum_{p \in L} W^i(p) \right\} + O(1), \tag{1}$$

where $A(L)$ is the number of bins used by algorithm A.

Consider the Super Harmonic algorithm. For $1 \leq i \leq k$, let s_i be the number of type i pieces. For $1 \leq i, s \leq k$, let $B_{(i)}$, $B_{(i,s)}$, $B_{(i,?)}$, $B_{(?,i)}$ be the number of bins in groups (i), (i, s), $(i, ?)$, and $(?, i)$. Then we have

$$\sum_i \left\{ B_{(i)} + \sum_s B_{(i,s)} + B_{(i,?)} \right\} = \sum_i \frac{(1 - \alpha_i)s_i}{\beta_i} + O(1) \tag{2}$$

and

$$\sum_i \left\{ B_{(?,i)} + \sum_s B_{(s,i)} \right\} = \sum_i \frac{\alpha_i s_i}{\gamma_i} + O(1). \tag{3}$$

So, for each item with size $x \in I_i$, where $i \leq k$, if we define its weight as:

$$\frac{1 - \alpha_i}{\beta_i} + \frac{\alpha_i}{\gamma_i},$$

then it is not difficult to see that the constraint (1) holds. However, this weighting function is not good enough, that is, it always leads to a competitive ratio of at least 1.69103 if we use this weighting function to analyze the Super Harmonic algorithm.

The main reason is that for each bin in group (i, s) we account it twice, where $1 \leq i, s \leq k$. Next, we give some intuitions for improving this weighting function.

By (2) and (3), observe that

$$\sum_i \sum_s B_{(i,s)} \leq \sum_i \frac{(1-\alpha_i)s_i}{\beta_i} + O(1) \tag{4}$$

and

$$\sum_i \sum_s B_{(s,i)} \leq \sum_i \frac{\alpha_i s_i}{\gamma_i} + O(1). \tag{5}$$

So, we have

$$\begin{aligned}\sum_i \sum_s B_{(i,s)} &= \frac{\sum_i \sum_s B_{(i,s)} + \sum_i \sum_s B_{(s,i)}}{2} \\ &\leq \sum_i \frac{(1-\alpha_i)s_i}{2\beta_i} + \sum_i \frac{\alpha_i s_i}{2\gamma_i} + O(1) \\ &= s_i \sum_i \left(\frac{1-\alpha_i}{2\beta_i} + \frac{\alpha_i}{2\gamma_i}\right) + O(1).\end{aligned}$$

Hence, for an item with size $x \in I_i$, after packing, if there is a bin in group (i, s) and also a bin in group (s, i), then we can define its weight as below:

$$\frac{1-\alpha_i}{2\beta_i} + \frac{\alpha_i}{2\gamma_i}.$$

This is the main intuition for defining our new weighting functions, which will be given in the next subsection.

3.2. New Weighting Functions

Remember that in Super Harmonic, there is a set $D = \{\Delta_0, \Delta_1, \ldots, \Delta_K\}$ representing the "free spaces" reserved for red items. Recall the two functions $\phi(i)$ and $\varphi(i)$ are related to free spaces and have the following meanings: $\phi(i) = j$ implies that free space Δ_j is reserved for red items in a bin consisting of β_i blue items of type i, and $\varphi(i) = j$ indicates that a red item of type i could be packed in free space $\Delta_{\geq j}$.

We are now ready to define new weighting functions. Items with size larger than ϵ will be first considered. Let E be the number of indeterminate red group bins $(?, i)$ when the whole packing is done.

If $E = 0$, that is, every red item is placed in a bin with one or more blue items, then we define the weighting function as:

$$W^1(x) = \frac{1-\alpha_i}{\beta_i} \qquad \text{if } x \in I_i. \tag{6}$$

Otherwise, $E > 0$ implies that for some i, an indeterminate red group bin $(?, i)$ exists after packing. Let r be the type of the smallest red item in the indeterminate red group

bins. If $2 \le \varphi(r) \le K$, then we define the corresponding weighting functions as follows:

$$W^{K+2-\varphi(r)}(x) = \begin{cases} \frac{1-\alpha_i}{\beta_i} + \frac{\alpha_i}{2\gamma_i} & \text{if } x \in I_i,\ \phi(i) < \varphi(r) \text{ and } \varphi(i) < \varphi(r), \\ \frac{1-\alpha_i}{\beta_i} + \frac{\alpha_i}{\gamma_i} & \text{if } x \in I_i,\ \phi(i) < \varphi(r) \text{ and } \varphi(i) \ge \varphi(r), \\ \frac{1-\alpha_i}{2\beta_i} + \frac{\alpha_i}{\gamma_i} & \text{if } x \in I_i,\ \phi(i) \ge \varphi(r) \text{ and } \varphi(i) \ge \varphi(r), \\ \frac{1-\alpha_i}{2\beta_i} + \frac{\alpha_i}{2\gamma_i} & \text{if } x \in I_i,\ \phi(i) \ge \varphi(r) \text{ and } \varphi(i) < \varphi(r). \end{cases}$$

If $\varphi(r) = 1$, we define

$$W^{K+1}(x) = \begin{cases} \frac{1-\alpha_i}{\beta_i} & \text{if } x \in I_i,\ \phi(i) = 0 \text{ and } \varphi(i) = 0, \\ \frac{1-\alpha_i}{\beta_i} + \frac{\alpha_i}{\gamma_i} & \text{if } x \in I_i,\ \phi(i) = 0 \text{ and } \varphi(i) > 0, \\ 0 & \text{if } x \in I_i,\ \phi(i) > 0 \text{ and } \varphi(i) = 0, \\ \frac{\alpha_i}{\gamma_i} & \text{if } x \in I_i,\ \phi(i) > 0 \text{ and } \varphi(i) > 0. \end{cases}$$

Note that in these definitions, if $\gamma_i = 0$ then we replace $\frac{\alpha_i}{\gamma_i}$ with zero. For an item with size $x \in I_{k+1}$, we always define $W^j(x) = x/(1-\epsilon)$ for all j.

THEOREM 3.1. *For any list L, we have*

$$A(L) \le \max_{1 \le i \le K+1} \left\{ \sum_{p \in L} W^i(p) \right\} + O(1),$$

where $A(L)$ is the number of bins used by Super Harmonic for list L.

PROOF. Fix a list L. Let D be the sum of the sizes of the items of type $(k+1)$. By NEXT FIT, we know that the number of bins used for type $(k+1)$ is at most $D/(1-\epsilon)+1$.

Again, we use E to denote the number of indeterminate red group bins when all the packing is done. For $1 \le i \le k$, let s_i be the number of type i pieces. Let $B_{(i)}$, $B_{(i,s)}$, $B_{(i,?)}$, $B_{(?,i)}$ be the number of bins in groups (i), (i,s), $(i,?)$ and $(?,i)$.

To prove this theorem, we consider two cases.

Case 1. If $E = 0$, that is, $\sum_i B_{(?,i)} = 0$, every red item is packed in a bin with one or more blue items. Therefore, we just need to count bins containing blue items:

$$\begin{aligned} A(L) &\le \frac{D}{1-\epsilon} + \sum_i \left\{ B_{(i)} + \sum_s B_{(i,s)} + B_{(i,?)} \right\} + O(1) \\ &\le \sum_{x \in I_{k+1}, x \in L} W^1(x) + \sum_i \frac{(1-\alpha_i)s_i}{\beta_i} + O(1) \quad \text{by (2)} \\ &= \sum_{x \in I_{k+1}, x \in L} W^1(x) + \sum_{x \notin I_{k+1}, x \in L} W^1(x) + O(1). \end{aligned}$$

Case 2. $E > 0$. We first bound the numbers of bins of $B_{(?,i)}$ and $B_{(i,?)}$ for all i, then consider two subcases to bound the total number of bins used.

Assume that $\varphi(r) = j \ge 1$ (by the definition of function φ). Let e be the smallest red item in indeterminate red group bins and r be its type. Then, every type i red item with $\varphi(i) < j$ is placed in a final group bin (s,i); otherwise, item e would not be the smallest

red item. Hence, we have

$$\sum_{\varphi(i)<j} B_{(?,i)} = 0. \tag{7}$$

On the other hand, we know a bin $B_{(?,r)}$ and a bin $B_{(i,?)}$ cannot exist at the same time by the rules of Super Harmonic if $\phi(i) \geq j$. Hence, we have

$$\sum_{\phi(i)\geq j} B_{(i,?)} = 0. \tag{8}$$

In accordance with the Super Harmonic algorithm, for any type bin $B_{(i)}$, we have

$$\phi(i) = 0. \tag{9}$$

Define

$$X = \sum_{\substack{\phi(i)\geq j \\ \varphi(s)<j}} B_{(i,s)},$$

which is the total number of all the bins in groups (i, s) such that $\phi(i) \geq j$ and $\varphi(s) < j$. Then, we have

$$\begin{aligned} A(L) &\leq \frac{D}{1-\epsilon} + \sum_i \left(B_{(i)} + B_{(i,?)} + B_{(?,i)}\right) + \sum_i \sum_s B_{(i,s)} + O(1) \\ &= \frac{D}{1-\epsilon} + \sum_i \left(B_{(i)} + B_{(i,?)} + B_{(?,i)}\right) + X + \sum_{\phi(i)<j} \sum_s B_{(i,s)} \\ &\quad + \sum_{\varphi(i)\geq j} \sum_s B_{(s,i)} + O(1) \\ &= \frac{D}{1-\epsilon} + \sum_{\phi(i)<j} \left(B_{(i)} + B_{(i,?)} + \sum_s B_{(i,s)}\right) \\ &\quad + \sum_{\varphi(i)\geq j} \left(B_{(?,i)} + \sum_s B_{(s,i)}\right) + X + O(1). \end{aligned} \tag{10}$$

The last inequality holds from equalities $\sum_i B_{(i)} = \sum_{\phi(i)<j} B_{(i)}$ by (9), $\sum_i B_{(i,?)} = \sum_{\phi(i)<j} B_{(i,?)}$ by (8) and $\sum_i B_{(?,i)} = \sum_{\varphi(i)\geq j} B_{(?,i)}$ by (7).

Case 2.1. $j \geq 2$. By the definition of variable X, we have

$$X \leq \sum_{j\leq\phi(i)\leq K} \sum_s B_{(i,s)} \text{ and } X \leq \sum_{1\leq\varphi(i)\leq j-1} \sum_s B_{(s,i)}.$$

It is not difficult to see,

$$X \leq \left\{ \sum_{j\leq\phi(i)\leq K} \sum_s B_{(i,s)} + \sum_{1\leq\varphi(i)\leq j-1} \sum_s B_{(s,i)} \right\} \Big/ 2. \tag{11}$$

Hence, by (10) and (11), we have

$$\begin{aligned}
A(L) &\le \frac{D}{1-\epsilon} + \sum_{\phi(i)<j}\left(B_{(i)} + B_{(i,?)} + \sum_s B_{(i,s)}\right) + \sum_{\varphi(i)\ge j}\left(B_{(?,i)} + \sum_s B_{(s,i)}\right) \\
&\quad + \sum_{\phi(i)\ge j}\sum_s \frac{B_{(i,s)}}{2} + \sum_{\varphi(i)<j}\sum_s \frac{B_{(s,i)}}{2} + O(1) \\
&\le \frac{D}{1-\epsilon} + \sum_{\phi(i)<j}\frac{(1-\alpha_i)s_i}{\beta_i} + \sum_{\varphi(i)\ge j}\frac{\alpha_i s_i}{\gamma_i} + \sum_{\phi(i)\ge j}\frac{(1-\alpha_i)s_i}{2\beta_i} + \sum_{\varphi(i)<j}\frac{\alpha_i s_i}{2\gamma_i} + O(1) \\
&\le \frac{D}{1-\epsilon} + \sum_{\substack{\phi(i)<j\\ \varphi(i)<j}}\left(\frac{(1-\alpha_i)s_i}{\beta_i} + \frac{\alpha_i s_i}{2\gamma_i}\right) + \sum_{\substack{\phi(i)<j\\ \varphi(i)\ge j}}\left(\frac{(1-\alpha_i)s_i}{\beta_i} + \frac{\alpha_i s_i}{\gamma_i}\right) \\
&\quad + \sum_{\substack{\phi(i)\ge j\\ \varphi(i)\ge j}}\left(\frac{(1-\alpha_i)s_i}{2\beta_i} + \frac{\alpha_i s_i}{\gamma_i}\right) + \sum_{\substack{\phi(i)\ge j\\ \varphi(i)<j}}\left(\frac{(1-\alpha_i)s_i}{2\beta_i} + \frac{\alpha_i s_i}{2\gamma_i}\right) + O(1) \\
&= \sum_{x\in I_{k+1}, x\in L} W^{K+2-j}(x) + \sum_{x\notin I_{k+1}, x\in L} W^{K+2-j}(x) + O(1),
\end{aligned}$$

where the second inequality follows directly from (2) and (3) and the last equality holds from the new weighting functions defined in Subsection 3.2.

Case 2.2. $j = 1$. In accordance with the Super Harmonic algorithm, for any type of bin (i, s), we have $\varphi(s) \ge 1$, where $1 \le i, s \le k$ and k is a parameter defined in Super Harmonic. Hence, there is no such bin (i, s) with $\varphi(s) < 1$, that is, $X = \emptyset$.

Then, by (10), we have

$$\begin{aligned}
A(L) &\le \frac{D}{1-\epsilon} + \sum_{\phi(i)<1}\left(B_{(i)} + B_{(i,?)} + \sum_s B_{(i,s)}\right) + \sum_{\varphi(i)\ge 1}\left(B_{(?,i)} + \sum_s B_{(s,i)}\right) + O(1) \\
&\le \frac{D}{1-\epsilon} + \sum_{\phi(i)=0}\frac{(1-\alpha_i)s_i}{\beta_i} + \sum_{\varphi(i)\ge 1}\frac{\alpha_i s_i}{\gamma_i} + O(1) \\
&\le \frac{D}{1-\epsilon} + \sum_{\substack{\phi(i)=0\\ \varphi(i)=0}}\frac{(1-\alpha_i)s_i}{\beta_i} + \sum_{\substack{\phi(i)=0\\ \varphi(i)>0}}\left(\frac{(1-\alpha_i)s_i}{\beta_i} + \frac{\alpha_i s_i}{\gamma_i}\right) + \sum_{\substack{\phi(i)>0\\ \varphi(i)>0}}\frac{\alpha_i s_i}{\gamma_i} + O(1) \\
&= \sum_{x\in I_{k+1}, x\in L} W^{K+1}(x) + \sum_{x\notin I_{k+1}, x\in L} W^{K+1}(x) + O(1),
\end{aligned}$$

where the last equality holds from the new weighting functions defined in Section 3.2.

Therefore, we have $A(L) \le \max_{1\le i\le K+1}\{\sum_{p\in L} W^i(p)\} + O(1)$. □

4. ALGORITHM $H \otimes SH+$ AND ITS ANALYSIS

In the section, we first review a class of online algorithms for two dimensional online bin packing, called $H \otimes C$ [Seiden and van Stee 2003]. Next we introduce a new instance of algorithm $H \otimes SH+$, where H is Harmonic and $SH+$ (Strange Harmonic+) is an instance of Super Harmonic. Then we propose some new techniques on how to

bound the total weight in a single bin, which is crucial to obtaining a better asymptotic competitive ratio for the $H \otimes C$ algorithm. Finally, we apply new weighting functions for $SH+$ to analyze the two-dimensional online bin-packing algorithm $H \otimes SH+$ and show its competitive ratio at most 2.5545, which implies that the new weighting functions work very well with the generalized approach of bounding the total weight in a single bin.

4.1. Algorithms $H \times C$ and $H \otimes C$

Now we review two-dimensional online bin-packing algorithms $H \times C$ and $H \otimes C$ [Seiden and van Stee 2003], where H is Harmonic and C is Super Harmonic.

Given an item $p = (w, h)$, $H \times C$ operates as follows.

(1) *Packing Items into Slices*. If $w \geq \epsilon$ then pack p into a slice of height 1 and width t_i by H (Harmonic algorithm), where $t_{i+1} < w \leq t_i$; else pack it into a slice of height 1 and width $\epsilon(1-\delta)^i$ by H (Harmonic algorithm), where $\epsilon(1-\delta)^{i+1} < w \leq \epsilon(1-\delta)^i$ and $\delta > 0$ is arbitrarily small.
(2) *Packing Slices into Bins*. When a new slice is required in the step, we allocate it from a bin using algorithm C.

$H \otimes C$ is a randomized algorithm, which operates as follows: before processing begins, we flip a fair coin. If the result is heads, then we run $H \times C$; otherwise, we run $C \times H$, that is, the roles of height and width are interchanged. Note that it is possible to derandomize $H \otimes C$ without increasing its performance ratio. For details, we refer to Seiden and van Stee [2003].

THEOREM 4.1. *If an online 1D bin-packing algorithm C has weighting functions $W_C^i(x)$ such that $C(L) \leq \max_i\{\sum_{x \in L} W_C^i(x)\} + O(1)$, then the cost by algorithm $H \otimes C$ for input L is at most*

$$\frac{1}{2(1-\delta)} \left(\max_i \left\{ \sum_{p \in L} W_{H \times C}^i(p) \right\} + \max_i \left\{ \sum_{p \in L} W_{C \times H}^i(p) \right\} \right) + O(1),$$

and the asymptotic competitive ratio of algorithm $H \otimes C$ is at most

$$\frac{1}{2(1-\delta)} \max_X \left(\max_i \left\{ \sum_{(x,y) \in X} W_H(x) W_C^i(y) \right\} + \max_i \left\{ \sum_{(x,y) \in X} W_H(y) W_C^i(x), \right\} \right),$$

where δ is a parameter defined in $H \otimes C$ algorithm and X is any set of items that fit in a single bin.

4.2. An instance of Super Harmonic $SH+$

As mentioned in Seiden [2002], it is a hard problem to find appropriate parameters in designing an instance of Super Harmonic, especially setting t_i. The parameters in

$SH+$ are found through a trial-and-error way and are defined as follows:

i	t_i	α_i	β_i	δ_i	$\phi(i)$	$\varphi(i)$	γ_i
1	1	0	1	0	0	0	0
2	0.706	0	1	0.294	1	0	0
3	0.657	0	1	0.343	2	0	0
4	0.647	0	1	0.353	3	0	0
5	0.625	0	1	0.375	4	0	0
6	0.6	0	1	0.4	5	0	0
7	0.58	0	1	0.42	6	0	0
8	0.5	0	2	0	0	0	0
9	0.42	0.162	2	0.16	0	6	1
10	0.4	0.192	2	0.2	0	5	1
11	0.375	0.2346	2	0.25	0	4	1
12	0.353	0.3004	2	0.294	1	3	1
13	0.343	0.3077	2	0.314	1	2	1
14	1/3	0	3	0	0	0	0
15	0.294	0.0816	3	0.118	0	1	1
16	1/4	0.186	4	0	0	1	1
17	1/5	0.092	5	0	0	1	1
18	1/6	0.1456	6	0	0	1	1
19	0.147	0.2162	6	0.118	0	1	2
20	1/7	0.1525	7	0	0	1	2
$21-49$	$1/(i-13)$	$ff(i)$	$i-13$	0	0	1	$\lfloor \Delta_1/t_i \rfloor$
50	1/37	0	37	0	0	0	0
51	1/38	0	$*$	$*$	$*$	$*$	$*$

$j=\phi(i)$	Δ_j	Red accepted
1	0.294	15..50
2	0.343	13, 15..50
3	0.353	12, 13, 15..50
4	0.375	11..13, 15..50
5	0.4	10..13, 15..50
6	0.42	9..13, 15..50

where $ff(i) = 1.35(50-i)/37(i-12)$.

Then, we have seven weighting functions for $SH+$, that is, W_C^i as defined in the last section, where $1 \le i \le 7$.

4.3. Previous Framework for Calculating Upper Bounds

In this section, we first introduce the previous framework for computing the upper bound of the competitive ratio of $H \otimes SH+$, then mention that the previous framework does not work well with the instance in the last section, that is, the previous framework does not lead to a better upper bound.

Let $p=(x,y)$ be an item. We define the following functions.

$$W^i_{H\times C}(p) = W_H(x)W^i_C(y),\ \ W^i_{C\times H}(p) = W_H(y)W^i_C(x),$$

and

$$W^{i,j}(x,y) = \frac{W_H(x)W^i_C(y) + W^j_C(x)W_H(y)}{2}.$$

Then, we can obtain an upper bound on the competitive ratio R of algorithm $H\otimes SH+$ as follows by Theorems 3.1 and 4.1, where X is any set of items that fit in a single bin.

$$\begin{aligned} R &\le \frac{1}{2(1-\delta)} \max_X \left(\max_{1\le i\le 7} \left\{ \sum_{p\in X} W^i_{H\times C}(p) \right\} + \max_{1\le i\le 7} \left\{ \sum_{p\in X} W^i_{C\times H}(p) \right\} \right) \\ &= \frac{1}{(1-\delta)} \max_{1\le i,j\le 7, X} \left\{ \sum_{p\in X} \left(W^i_{H\times C}(p) + W^j_{C\times H}(p) \right)/2 \right\} \\ &= \frac{1}{(1-\delta)} \max_{1\le i,j\le 7, X} \left\{ \sum_{p\in X} W^{i,j}(x,y) \right\}. \end{aligned} \tag{12}$$

The value of R can be estimated by the following approach.

Definition 4.2. Let f be a function mapping from $(0, 1]$ to $\mathbb{R}^+$. $\mathcal{P}(f)$ is the mathematical program: maximize $\sum_{x\in X} f(x)$ subject to $\sum_{x\in X} \leq 1$, over all finite sets of real numbers X. We also use $\mathcal{P}(f)$ to denote the value of this mathematical program.

LEMMA 4.3 ([SEIDEN AND VAN STEE 2003]). *Let f and g be functions mapping from $(0, 1]$ to $\mathbb{R}^+$. Let $F = \mathcal{P}(f)$ and $G = \mathcal{P}(g)$. Then, the maximum of $\sum_{p\in X} f(h(p))g(w(p))$ over all finite multisets of items X that fit in a single bin is at most FG, where p is a rectangle and $h(p)$ and $w(p)$ are its height and width, respectively.*

In Seiden and van Stee [2003], f and g are defined as below:

$$f^{i,j}(y) = \frac{W_H(y) + W_C^i(y)}{2},$$

and

$$g^{i,j}(x) = \sup_{0<y\leq 1} \frac{W^{i,j}(x, y)}{f^{i,j}(y)}.$$

By these definitions, we have

$$W^{i,j}(x, y) \leq f^{i,j}(y)g^{i,j}(x), \tag{13}$$

for all $0 \leq x \leq 1$ and $0 \leq y \leq 1$.

4.4. A New Framework for Calculating Upper Bound

In this section, we first generalize the previous analysis framework by introducing a new lemma and developing new functions for f and g in order to bound the total weight in a single bin. Then we apply our new weighting functions for Super Harmonic to algorithm $H \otimes SH+$ and obtain a new upper bound for two-dimensional online bin packing.

LEMMA 4.4. $\max_X\{\sum_{p\in X} W^{i,j}(x, y)\} = \max_X\{\sum_{p\in X} W^{j,i}(x, y)\}$, *where $1 \leq i, j \leq 7$ and X is any feasible set for one bin.*

PROOF. By definition, observe that for any $1 \leq i, j \leq 7$,

$$W^{i,j}(x, y) = W^{j,i}(y, x). \tag{14}$$

Let $X = \{p_1, p_2, \ldots, p_m\}$ be a set of rectangles which fit into a single bin, where $p_i = (x_i, y_i)$ is the i-th rectangle in X. If we exchange roles of x and y of p_i to get new rectangles $p'_i = (y_i, x_i)$ for all i, then it is not difficult to see that the new set $X' = \{p'_1, p'_2, \ldots, p'_m\}$ is also a feasible pattern, that is, all items can fit in a single bin. On the other hand, by Eq. (14), we have

$$\sum_{p\in X} W^{i,j}(p) = \sum_{p'\in X'} W^{j,i}(p'),$$

where $1 \leq i, j \leq 7$. There is a one-to-one mapping between X and X' in all the feasible patterns. Therefore, we have this lemma. □

New Functions f and g. We give new functions f and g such that (i) Lemma 4.3 can be applied to bound the weight in a single bin, and (ii) the resultant bound is not too loose. The new functions f and g are given as follows:

$$f^{i,j}(y) = \lambda_{i,j} W_H(y) + (1 - \lambda_{i,j}) W_C^i(y),$$

where $0 \le \lambda_{i,j} \le 1$. By the same approach used in Seiden and van Stee [2003], function g is defined here:

$$g^{i,j}(x) = \sup_{0<y\le 1} \frac{W^{i,j}(x,y)}{f^{i,j}(y)}.$$

Note that in Seiden and van Stee [2003], $\lambda_{i,j}$ are 1/2 for all i, j. It is not difficult to see that (13) still holds for all $0 \le x \le 1$ and $0 \le y \le 1$.

New Approach of Calculating $\mathcal{P}(f)$. In order to use Lemma 4.3 to obtain the upper bound on the competitive ratio R of algorithm $H \otimes C$, we need to calculate $\mathcal{P}(f^{i,j})$ and $\mathcal{P}(g^{i,j})$. Let f be one of $f^{i,j}$ or $g^{i,j}$ for $1 \le i, j \le 7$. Seiden [2002] wrote a programming to enumerate all the feasible patterns to get the bounds for $\mathcal{P}(f)$. Here, we give a simple approach by calling LP solver directly to estimate $\mathcal{P}(f)$, which can be modeled as the following mixed integer program (MIP):

$$\begin{aligned} \text{max.} \quad & f = \sum_{i=1}^{50} x_i w_i + \left(1 - \sum_{i=1}^{50} x_i(t_{i+1}+\epsilon)\right) \times \frac{1}{1-t_{51}} \qquad (1) \\ \text{such that} \quad & \sum_{i=1}^{50} x_i(t_{i+1}+\epsilon) \le 1, \\ & x_i \ge 0, \text{ integer.} \end{aligned}$$

where x_i is the number of type i items in a feasible pattern, w_i is the weight for an item of type i, which is decided by function f, that is, $w_i = f(p)$ if $p \in (t_{i+1}, t_i]$. Note t_i is defined in Section 4.2. We can solve the above MIP (1) by setting $\epsilon = 0$. However, this approach does not give a good approximation solution since some infeasible solutions are involved and affect the optimal solution too much. In order to reduce the error caused by some bad infeasible solutions, we append some constraints that will help us to remove some bad infeasible solutions and do not eliminate any feasible solution of MIP (1). For example, a type i item has size larger than 0.5 if $1 \le i \le 7$, then at most one of type i can be accepted in one bin, that is, $x_i \le 1$ for $1 \le i \le 7$. Similarly, we have the following inequalities:

$$\begin{aligned} x_i &\le 2, \quad \text{for } 8 \le i \le 13, \\ x_i &\le 3, \quad \text{for } 14 \le i \le 15, \\ x_i &\le i-12, \quad \text{for } 16 \le i \le 17, \\ x_{18}+x_{19} &\le 6, \\ x_i &\le i-13, \quad \text{for } 20 \le i \le 50. \end{aligned}$$

Moreover, we can add some complicated constraints such as

$$\begin{aligned} 2x_7 + x_{15} &\le 3.9, \\ 3x_7 + 2x_{13} + x_{17} &\le 5.9, \\ 4x_{13} + 3x_{15} + x_{24} &\le 11.9, \\ 5x_7 + 3.53x_{11} + 1.47x_{18} &\le 9, \\ 12x_7 + 8x_{13} + 3x_{20} + x_{36} &\le 23, \\ 9x_7 + 6x_{13} + 2x_{21} + x_{30} &\le 17. \end{aligned}$$

This constraints do not eliminate any feasible solution of MIP (1). For example, consider the constraint $5x_7 + 3.53x_{11} + 1.47x_{18} \le 9$. Since an item of type 7 has size larger than 0.5, an item of type 11 has size larger than 0.353 and an item of type 18 has size

larger than 0.147, we have $0.5x_7 + 0.353x_{11} + 0.147x_{18} < 1$. Equivalently, we have $5x_7 + 3.53x_{11} + 1.47x_{18} < 10$. It is not difficult to see that the inequality $5x_7 + 3.53x_{11} + 1.47x_{18} \leq 9$ is equivalent to $5x_7 + 3.53x_{11} + 1.47x_{18} < 10$ when x_7, x_{11} and x_{18} are nonnegative integers. For other constraints, the arguments are analogous.

Therefore, we remodel this MIP as follows.

$$\text{max.} \qquad f = \sum_{i=1}^{50} x_i w_i + \left(1 - \sum_{i=1}^{50} x_i t_{i+1}\right) \times \frac{1}{1 - t_{51}} \qquad (2)$$

$$\text{subject to} \quad \sum_{i=1}^{50} x_i t_{i+1} \leq 1,$$

$$\begin{array}{ll}
x_i \leq 1, \text{ for } 1 \leq i \leq 7, & x_i \leq 2, \text{ for } 8 \leq i \leq 13, \\
x_i \leq 3, \text{ for } 14 \leq i \leq 15, & x_i \leq i - 12, \text{ for } 16 \leq i \leq 17, \\
x_{18} + x_{19} \leq 6, & x_i \leq i - 13, \text{ for } 20 \leq i \leq 50, \\
2x_7 + x_{15} \leq 3.9, & 3x_7 + 2x_{13} + x_{17} \leq 5.9, \\
4x_{13} + 3x_{15} + x_{24} \leq 11.9, & 5x_7 + 3.53x_{11} + 1.47x_{18} \leq 9, \\
12x_7 + 8x_{13} + 3x_{20} + x_{36} \leq 23, & 9x_7 + 6x_{13} + 2x_{21} + x_{30} \leq 17, \\
x_i \geq 0, \text{ integer.} &
\end{array}$$

To solve MIP (2), we use a tool for solving linear and integer programs called *GLPK* [GLP]. We write a program to calculate $W^{i,j}(x, y)$, $g^{i,j}(x)$ and $f^{i,j}(y)$ for each (i, j), and then call the API of GLPK to calculate $\mathcal{P}(f^{i,j})$ and $\mathcal{P}(g^{i,j})$. The values of $\mathcal{P}(f^{i,j})$ and $\mathcal{P}(g^{i,j})$ are shown in the tables in Appendix.

Note that when we use Lemma 4.3 for the upper bound on the weight $\max_{\forall X}\{\sum_{p\in X} W^{i,j}(x, y)\}$, for all pairs (i, j), the calculations are independent. For different pairs (i, j), $\lambda_{i,j}$ may be different. So, in order to get an upper bound near the true value of $\max_{\forall X}\{\sum_{p\in X} W^{i,j}(x, y)\}$, we have to select an appropriate $\lambda_{i,j}$. This can be done by a trial-and-error approach.

THEOREM 4.5. *For all $\delta > 0$, the asymptotic competitive ratio of $H \otimes B$ is at most 2.5545.*

PROOF. In accordance with the tables in Appendix, by Lemma 4.4 and Lemma 4.3, we have

$$\max_{\forall X}\left\{\sum_{p\in X} W^{1,2}(p)\right\} = \max_{\forall X}\left\{\sum_{p\in X} W^{2,1}(p)\right\} \leq \mathcal{P}(f^{1,2})\mathcal{P}(g^{1,2}) \leq 2.5539.$$

$$\max_{\forall X}\left\{\sum_{p\in X} W^{1,6}(p)\right\} = \max_{\forall X}\left\{\sum_{p\in X} W^{6,1}(p)\right\} \leq \mathcal{P}(f^{6,1})\mathcal{P}(g^{6,1}) \leq 2.5545.$$

$$\max_{\forall X}\left\{\sum_{p\in X} W^{2,5}(p)\right\} = \max_{\forall X}\left\{\sum_{p\in X} W^{5,2}(p)\right\} \leq \mathcal{P}(f^{5,2})\mathcal{P}(g^{5,2}) \leq 2.5340.$$

$$\max_{\forall X}\left\{\sum_{p\in X} W^{2,6}(p)\right\} = \max_{\forall X}\left\{\sum_{p\in X} W^{6,2}(p)\right\} \leq \mathcal{P}(f^{6,2})\mathcal{P}(g^{6,2}) \leq 2.5364.$$

For all the other (i, j), by Lemma 4.3, we have

$$\max_{\forall X}\left\{\sum_{p\in X} W^{i,j}(p)\right\} \le \mathcal{P}(f^{i,j})\mathcal{P}(g^{i,j}) \le \mathcal{P}(f^{1,1})\mathcal{P}(g^{1,1}) \le 2.5545. \qquad \square$$

Remarks. If we use the weighting functions from Seiden [2002] and the previous analysis framework, we find that the competitive ratio is at least 3.04. (run our programming 2DHSH.c like "./2DHSH+.exe old > yourfile") Even if we use the new weighting function, by the previous analysis framework, the competitive ratio is still at least 3.04, by running our programming 2DHSH.c like "./2DHSH+.exe new1 > yourfile". We also find that if we use the old weighting function from Seiden [2002] with the new analysis framework, the competitive ratio is at least 2.79. (run our programming 2DHSH.c like "./2DHSH+.exe old2 > yourfile") The reason is that Lemma 4.3 does not work very well with the old weight function, that is, the resulting value $F \cdot G$ is away from the maximum weight of items in a single bin.

5. CONCLUDING REMARKS

When the parameters in Super Harmonic such as α_i, β_i, γ_i and $\phi(i)$ and $\varphi(i)$ are given, we can calculate the weighting functions of Super Harmonic $W_B^j(\cdot)$. Then, the weighting functions $W^{i,j}(x, y)$ for algorithm $H \otimes SH+$ can be calculated as well as $f^{i,j}(y)$ and $g^{i,j}(x)$. For each (i, j), we call the API of GLPK to solve $\mathcal{P}(f^{i,j})$ and $\mathcal{P}(g^{i,j})$.

To use our program under *linux* system:

—Install GLPK,
—Compile: "gcc -o 2DHSH+.exe 2DHSH+.c -lglpk"
—Run: "./2DHSH+.exe new2 > yourfile"

If there is an error message like "Could not load *.so" when you compile the source, then try to set "LD_LIBRARY_PATH" as follows: "LD_LIBRARY_PATH= $LD_LIBRARY_PATH:/usr/local/lib", then "export LD_LIBRARY_PATH".

When we use the tool for solving the mixed integer programs, there are two files which are necessary: one is the model file for the linear or integer program itself (refer to Appendix), and the other is the data file where the data is stored. We write a program to generate the data and then call the tool GLPK. (Actually, we call the API (Application Program Interface) of GLPK. To download the source file, go to: http://sites.google.com/site/xinhan2009/Home/files/2DHSH.c).

Our framework can be applied to 3D online bin packing to result in an algorithm $H \times H \otimes SH+$ with its competitive ratio $2.5545 \times 1.69103 (\approx 4.3198)$.

APPENDIXES

A. VALUES OF $F^{I,J}$ AND $G^{I,J}$

$(i, j) =$	$(1, 1)$	$(1, 2)$	$(1, 3)$	$(1, 4)$	$(1, 5)$	$(1, 6)$	$(1, 7)$
$\lambda_{i,j}$	0.500000	0.500000	0.540000	0.550000	0.565000	0.565000	0.600000
$\mathcal{P}(f^{i,j})$	1.598272	1.598272	1.605095	1.606845	1.609490	1.609490	1.615665
$\mathcal{P}(g^{i,j})$	1.598272	1.597872	1.574422	1.581742	1.585430	1.587508	1.575580
$\mathcal{P}(f^{i,j})\mathcal{P}(g^{i,j})$	2.554474	2.553834	2.527096	2.541614	2.551734	2.555079	2.545610

$(i, j) =$	$(2, 1)$	$(2, 2)$	$(2, 3)$	$(2, 4)$	$(2, 5)$	$(2, 6)$	$(2, 7)$
$\lambda_{i,j}$	0.500000	0.500000	0.530000	0.550000	0.565000	0.565000	0.600000
$\mathcal{P}(f^{i,j})$	1.597328	1.597328	1.597148	1.597028	1.596938	1.596938	1.596729
$\mathcal{P}(g^{i,j})$	1.609235	1.598326	1.586301	1.595016	1.602278	1.604268	1.589545
$\mathcal{P}(f^{i,j})\mathcal{P}(g^{i,j})$	2.570476	2.553051	2.533557	2.547285	2.558739	2.561917	2.538073

$(i,j)=$	$(3,1)$	$(3,2)$	$(3,3)$	$(3,4)$	$(3,5)$	$(3,6)$	$(3,7)$
$\lambda_{i,j}$	0.500000	0.500000	0.530000	0.550000	0.565000	0.565000	0.600000
$\mathcal{P}(f^{i,j})$	1.573676	1.573676	1.572837	1.572777	1.572732	1.572732	1.572627
$\mathcal{P}(g^{i,j})$	1.609235	1.598326	1.586301	1.595016	1.602278	1.604268	1.589545
$\mathcal{P}(f^{i,j})\mathcal{P}(g^{i,j})$	2.532414	2.515247	2.494992	2.508604	2.519954	2.523084	2.499762

$(i,j)=$	$(4,1)$	$(4,2)$	$(4,3)$	$(4,4)$	$(4,5)$	$(4,6)$	$(4,7)$
$\lambda_{i,j}$	0.500000	0.500000	0.535000	0.550000	0.565000	0.565000	0.600000
$\mathcal{P}(f^{i,j})$	1.581245	1.581245	1.577140	1.575380	1.573621	1.573621	1.569515
$\mathcal{P}(g^{i,j})$	1.609235	1.598326	1.586855	1.595016	1.602278	1.604268	1.589545
$\mathcal{P}(f^{i,j})\mathcal{P}(g^{i,j})$	2.544594	2.527344	2.502692	2.512755	2.521378	2.524510	2.494814

$(i,j)=$	$(5,1)$	$(5,2)$	$(5,3)$	$(5,4)$	$(5,5)$	$(5,6)$	$(5,7)$
$\lambda_{i,j}$	0.500000	0.500000	0.535000	0.550000	0.565000	0.565000	0.600000
$\mathcal{P}(f^{i,j})$	1.585370	1.585370	1.580113	1.577860	1.575607	1.575607	1.570350
$\mathcal{P}(g^{i,j})$	1.609542	1.598326	1.587240	1.595374	1.602747	1.604737	1.589740
$\mathcal{P}(f^{i,j})\mathcal{P}(g^{i,j})$	2.551720	2.533939	2.508019	2.517277	2.525300	2.528436	2.496449

$(i,j)=$	$(6,1)$	$(6,2)$	$(6,3)$	$(6,4)$	$(6,5)$	$(6,6)$	$(6,7)$
$\lambda_{i,j}$	0.500000	0.500000	0.530000	0.550000	0.565000	0.565000	0.600000
$\mathcal{P}(f^{i,j})$	1.586853	1.586853	1.582237	1.579160	1.576853	1.576853	1.571468
$\mathcal{P}(g^{i,j})$	1.609785	1.598326	1.586682	1.595657	1.603117	1.605107	1.589894
$\mathcal{P}(f^{i,j})\mathcal{P}(g^{i,j})$	2.554493	2.536309	2.510507	2.519798	2.527881	2.531019	2.498468

$(i,j)=$	$(7,1)$	$(7,2)$	$(7,3)$	$(7,4)$	$(7,5)$	$(7,6)$	$(7,7)$
$\lambda_{i,j}$	0.500000	0.515000	0.535000	0.555000	0.565000	0.570000	0.600000
$\mathcal{P}(f^{i,j})$	1.568686	1.560602	1.549821	1.539044	1.533655	1.530958	1.517143
$\mathcal{P}(g^{i,j})$	1.621572	1.609605	1.602462	1.612258	1.622822	1.638219	1.624359
$\mathcal{P}(f^{i,j})\mathcal{P}(g^{i,j})$	2.543738	2.511952	2.483529	2.481335	2.488849	2.508043	2.464386

B. MODEL FILE FOR GLPK AND USAGE OF OUR PROGRAM 2DHSH+.C

```
param I:=50;

param c{i in 1..I}>=0;

param w{i in 1..I};

var x{i in 1..I}, integer, >=0;

maximize f: sum{i in 1..I} w[i]*x[i] + (1-sum{i in 1..I} c[i]*x[i]) * 38/37;

s.t.    x0: sum{i in 1..I} c[i]*x[i] <= 1;
        x1: sum{i in 1..7} x[i] <= 1;
        x7: sum{i in 8..13} x[i] <= 2;
        x14: x[14] <= 3;
        x15: x[15] <= 3;

        x16: x[16] <= 4;
        x17: x[17] <= 5;
        x18: x[18] + x[19] <= 6;
        y715: 2*x[7]  + x[15]  <= 3.9;
        y71317: 3*x[7] + 2*x[13] + x[17] <= 5.9;
```

```
        y131524: 4*x[13] + 3*x[15] + x[24] <= 11.9;
        y71118:  5*x[7] + 3.53*x[11] + 1.47 *x[18] <= 9;
        y7132036: 12*x[7]+8*x[13] + 3*x[20] + x[36] <=23;
        y7132130: 9*x[7] + 6*x[13] + 2*x[21] + x[30] <=17;
         others{i in 20..50}: x[i] <= i -13;
end;
```

ACKNOWLEDGMENTS

The authors wish to thank the referees for their useful comments on the earlier draft of the article. Their suggestions have helped improve the presentation of the paper.

REFERENCES

BANSAL, N., CAPRARA, A., AND SVIRIDENKO, M. 2009. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM J. Comput. 39,* 4, 1256–1278.

BANSAL, N., CORREA, J. R., KENYON, C., AND SVIRIDENKO, M. 2006. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Math. Oper. Res. 31,* 1, 31–49.

BLITZ, D., VAN VLIET, A., AND WOEGINGER, G. J. 1996. Lower bounds on the asymptotic worst-case ratio of online bin packing alorithms. Unpublished manuscript.

BROWN, D. 1979. A lower bound for on-line one-dimensional bin packing algorithms. Tech. rep. R864, Coordinated Science Lab., Urbana, Illinois.

CAPRARA, A. 2002. Packing 2-dimensional bins in harmony. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 490–499.

CHUNG, F., GAREY, M., AND JOHNSON, D. 1982. On packing two-dimensional bins. *SIAM J. Alg. Disc. Meth. 3,* 1, 66–76.

COFFMAN, E., GAREY, M., AND JOHNSON, D. 1987. *Approximation Algorithms for Bin Packing: A Survey, chapter 2*. PWS, Boston, MA.

COPPERSMITH, D. AND PAGHAVAN, P. 1989. Multidimensional on-line bin packing: Algorithms and worst case analysis. *Oper. Res. Lett 8*, 17–20.

CSIRIK, J., FRENK, J. B. G., AND LABBÉ, M. 1993. Two-dimensional rectangle packing: On-line methods and results. *Disc. Appl. Math. 45,* 3, 197–204.

CSIRIK, J. AND VAN VLIET, A. 1993. An on-line algorithm for multidimensional bin packing. *Oper. Res. Lett 13*, 149–158.

EPSTEIN, L. AND VAN STEE, R. 2004. Optimal online bounded space multidimensional packing. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*. 214–223.

EPSTEIN, L. AND VAN STEE, R. 2005a. Online square and cube packing. *Acta Inf. 41,* 9.

EPSTEIN, L. AND VAN STEE, R. 2005b. Optimal online algorithms for multidimensional packing problems. *SIAM J. Comput. 35,* 2, 431–448.

GALAMBOS, G. 1991. A 1.6 lower-bound for the two-dimensional on-line rectange bin-packing. *Acta Cybern. 10,* 1-2, 21–24.

GALAMBOS, G. AND VAN VLIET, A. 1994. Lower bounds for 1-, 2- and 3-dimensional on-line bin packing algorithms. *Computing 52,* 3, 281–297.

GLP. http://www.gnu.org/software/glpk.

HAN, X., FUJITA, S., AND GUO, H. 2001. A two-dimensional harmonic algorithm with performance ratio 2.7834. *IPSJ SIG Notes 93*, 43–50.

HAN, X., YE, D., AND ZHOU, Y. 2006. Improved online hypercube packing. In *Proceedings of the 4th Workshop on Approximation and Online Algorithms (WAOA)*. 226–239.

JOHNSON, D., DEMERS, A., ULLMAN, J., GAREY, M., AND GRAHAM, R. 1974. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput. 3,* 4, 299–325.

LEE, C. AND LEE, D. 1985. A simple on-line packing algorithm. *J. ACM 32*, 562–572.

LIANG, F. 1980. A lower bound for online bin packing. *Inf. Proc. Lett. 10*, 76–79.

MIYAZAWA, F. AND WAKABAYASHI, Y. 2003. Cube packing. *Theoret. Comput. Sci. 297,* 1–3, 355–366.

RAMANAN, P., BROWN, D., LEE, C., AND LEE, D. 1989. On-line bin packing in linear time. *J. Algor. 10*, 305–326.

SEIDEN, S. 2002. On the online bin packing. *J. ACM 49*, 640–671.

SEIDEN, S. AND VAN STEE, R. 2003. New bounds for multidimensional packing. *Algorithmica 36*, 261–293.

VAN VLIET, A. 1992. An improved lower bound for on-line bin packing algorithms. *Inf. Proc. Letters. 43*, 277–284.

VAN VLIET, A. 1995. Lower and upper bounds for online bin packing and scheduling heuristics. Ph.D. dissertation. Erasmus University.

YAO, A.-C. 1980. New algorithms for bin. *J. ACM 27*, 207–227.

Received October 2008; revised April 2009, April 2010, September 2010; accepted October 2010