An Asymptotic Competitive Scheme for Online Bin Packing

Lin Chen, Deshi Ye, and Guochuan Zhang(✉)

College of Computer Science, Zhejiang University, Hangzhou 310027, China
{yedeshi,zgc}@zju.edu.cn

Abstract. We study the online bin packing problem, in which a list of items with integral size between 1 to B arrives one at a time. Each item must be assigned in a bin of capacity B upon its arrival without any information on the next items, and the goal is to minimize the number of used bins. We present an asymptotic competitive scheme, i.e., for any $\epsilon > 0$, the asymptotic competitive ratio is at most $\rho^* + \epsilon$, where ρ^* is the smallest possible asymptotic competitive ratio among all online algorithms.

Keywords: Online algorithms · Competitive scheme · Bin packing

1 Introduction

Bin packing is one of the well-known combinatorial optimization problems in operations research and theoretical computer science. An instance of bin packing consists of a set of items with integral size up to B (a given integer), and the goal is to pack these items into a minimum number of bins of size B. The offline bin packing problem, where all items are available before packing starts, is NP-hard [7]. In terms of asymptotic performance ratio, a standard measure for bin packing algorithms, de la Vega and Lueker [6] presented an APTAS and Karmakar and Karp [11] improved this result by giving an AFPTAS. Apart from this classical model, one can find many interesting extensions (e.g., [2,17]).

In the scenario of online bin packing, items arrive one by one in a list. Upon arrival of an item it must be irrevocably packed into a bin without knowing the subsequent items. Given an instance I, let $A(I)$ and $OPT(I)$ be the number of bins used by an online algorithm A and the optimal number of bins needed, respectively. The *asymptotic competitive ratio* ρ_A^∞ of algorithm A is the infimum ρ such that the following inequality holds for any instance I, where κ is a constant,

$$A(I) \leq \rho OPT(I) + \kappa.$$

One of the first online bin packing algorithms, *First Fit*, was studied by Ullman and Johnson et al. [9,15]. They proved that the asymptotic competitive

Research was supported in part by NSFC(11071215,11271325).

Z. Zhang et al. (Eds.): COCOA 2014, LNCS 8881, pp. 13–24, 2014.
DOI: 10.1007/978-3-319-12691-3_2

ratio of First Fit is 1.7. Then a sequence of improvements was proposed [12,13,16] and the currently best known upper bound is 1.58889 [14], while the best known lower bound is 1.54037 [1]. Very recently, the *competitive ratio approximate scheme* was introduced to online parallel machine scheduling problems by Günther et al. [8]. For any given $\epsilon > 0$, there exists an online algorithm $\{A_\epsilon\}$ that achieves a competitive ratio at most of $(1+\epsilon)$ times the optimal competitive ratio. Motivated by their work, we revisit the online bin packing problem. Following the simplified notion as [4], we use the *competitive scheme* instead of the *competitive ratio approximation scheme* in this paper. Our task is to design an asymptotic competitive scheme for the online bin packing problem. For simplicity, throughout the paper, we use competitive ratio instead of asymptotic competitive ratio.

Our Contribution. Let ρ^* be the competitive ratio of a best possible online algorithm. We show the following result.

Theorem 1. *The online bin packing problem admits an asymptotic competitive scheme* $\{A_{\epsilon,\kappa}|\epsilon > 0\}$ *satisfying that* $A_{\epsilon,\kappa}(I) \leq (\rho^* + O(\epsilon))OPT(I) + \kappa$*, where* κ *and* ϵ *are constants, and the running time of* $A_{\epsilon,\kappa}$ *is polynomial if* B *is fixed.*

General Idea. To prove Theorem 1, we start with the *bounded instances* where the adversary only releases a constant number of items. Indeed, if the adversary only releases C items, then the number of all the possible sequences of items is bounded by B^C, which is also a constant. It is not difficult to imagine that a best possible online algorithm for the bounded instances could be determined. Suppose this algorithm has a competitive ratio of ρ_0, then $\rho^* \geq \rho_0$ since even if we restrict the adversary to release at most C items, no online algorithm has a competitive ratio better than ρ_0. The main technical part is to show that, once C is large enough, we can generalize the algorithm of competitive ratio ρ_0 for bounded instances to an algorithm of competitive ratio $\rho_0 + O(\epsilon)$ for the general instances. To this end, we introduce the notion of *modified instances* as an intermediate. In a modified instance, the adversary can release an arbitrary number of items, however, the item list must conform to a certain pattern. We will show that, an online algorithm for bounded instances could be generalized to an online algorithm for modified instances with a loss of $O(\epsilon)$ in its competitive ratio. Meanwhile, an online algorithm for modified instances could also be generalized to an online algorithm for general instances with a loss of $O(\epsilon)$ in its competitive ratio.

The paper is organized as follows. In Sect. 2, we provide some definitions and notations. In Sect. 3, we show how to derive a best possible algorithm for the bounded instances. It remains to show how the algorithm for bounded instances could be generalized to an algorithm for modified instances, which is further generalized to an algorithm for general instances. The latter part is easier and we address it in Sect. 4, while the former part is presented in Sect. 5.

2 Preliminaries

Given the bin size B, an input of the online bin packing problem is a list (sequence) of items $(J_1, J_2, \ldots, J_n)$ for $n > 0$, where the i-th item is denoted by J_i, and we abuse the notation J_i to denote the size of the i-th item, which is an integer belonging to $\{1, 2, \cdots, B\}$. Given n items as an input, any packing of these n items into (at most n) bins could be represented by a $(2B)$-tuple $(r(n), x(n))$, where

- $r(n) = (r_1(n), r_2(n), \ldots, r_B(n))$, where $r_i(n)$ is the number of items of size exactly i;
- $x(n) = (x_1(n), x_2(n), \ldots, x_B(n))$, where $x_i(n)$ is the number of bins whose free space is exactly $B - i$ for $1 \le i \le B$.

Obviously, $\sum_{i=1}^{B} r_i(n) = n$, and the number of bins used is $\sum_{i=1}^{B} x_i(n)$. We call $(r(n), x(n))$ as a *state* and write $\eta^n = (r(n), x(n))$. If it is clear from context, we also write $(r(n), x(n))$ as (r, x) for simplicity. Let ST_n be the set of all the states with n items (i.e., all possible $(r(n), x(n))$'s), and denote its cardinality as $|ST_n|$. We can thus list these states as $\eta_1^n, \cdots, \eta_{|ST_n|}^n$. Specifically, we will use η^n to denote an arbitrary state in ST_n. Note that ST_0 consists of a unique state $\eta_1^0 = (0, 0, \cdots, 0)$.

Given any state $\eta^n = (r(n), x(n))$, we denote as $OPT(r(n))$ the optimal number of bins used when the items of $r(n)$ are packed. As a consequence, we define the *instant ratio* of the state η^n as

$$\tilde{\rho}(\eta^n) = \max\{1, (\sum_{i=1}^{B} x_i(n) - \kappa)/OPT(r(n))\}.$$

Specifically, define $\tilde{\rho}(\eta_1^0) = 1$. Here the constant κ in the above definition is the κ in Theorem 1.

We can interpret an online algorithm for the bin packing problem in terms of the states. Indeed, when an algorithm is applied to an item list $(J_1, J_2, \ldots, J_n)$, it returns a list of states $\eta^0 \to \eta^1 \to \cdots \to \eta^n$, where η^i is the state in which the first i items are packed. Specifically, if the competitive ratio of this algorithm is ρ, then $\tilde{\rho}(\eta^i) \le \rho$ for any i, and meanwhile there exists a certain item list $(J_1^*, J_2^*, \ldots, J_n^*)$ such that $\tilde{\rho}(\eta^n) = \rho$. In this view, the competitive ratio of an online algorithm is the instant ratio of the worst state it could ever return.

Recall that the Next-Fit algorithm [10] for bin packing has a competitive ratio of 2 (both in terms of asymptotic competitive ratio and absolute competitive ratio). Thus $\rho^* \le 2$ and we focus on states with instant ratio no more than 2. States with instant ratio larger than 2 are deleted beforehand. Let d be some constant that will be specified later and $R = ST_d$ for simplicity. For any integer $k > 0$, we define

$$kR = \{(kr(d), kx(d)) = (kr_1(d), \cdots, kr_B(d), kx_1(d), \cdots, kx_B(d)) | (r(d), x(d)) \in R\}.$$

Obviously, $kR \subset ST_{kd}$. A state $(\hat{r}(kd), \hat{x}(kd)) \in ST_{kd}$ is called a *neighbor* of $(kr(d), kx(d)) \in kR$ if $|\hat{r}_i(kd) - kr_i(d)| < k$ and $|\hat{x}_i(kd) - kx_i(d)| < k$

for all i. According to this definition, a state in ST_{kd} might be the neighbor of multiple states of kR. To make the notion of 'neighborhood' unique, we define an *assignment* as a mapping that assigns every state in ST_{kd} to be a neighbor of a unique state in kR (which can be achieved by assigning every state in ST_{kd} to an arbitrary one of its neighbors). Given an assignment, all the states in ST_{kd} are divided into $|R|$ disjoint sets, with each containing one state of kR and all its neighbors. Finally we define the *perturbation*. A perturbation is a vector $\Delta = (\Delta(r), \Delta(x))$, where $\Delta(r) = (\Delta_1(r), \cdots, \Delta_B(r))$, $\Delta(x) = (\Delta_1(x), \cdots, \Delta_B(x))$ with each coordinate being an integer. We define $D = ||\Delta||_\infty = \max\{|\Delta_i(r)|, |\Delta_i(x)|\}$, and write $(r', x') = (r, x) + \Delta$ as the normal vector addition. It is not difficult to verify that if $OPT(r) > BD$, then

$$\tilde{\rho}(r', x') \leq \frac{\sum x_i + BD}{OPT(r) - BD}.$$

The above formula is useful in characterizing how a slight perturbation will change the instant ratio of a state.

3 Bounded Instances

We consider bounded instances of bin packing, where the bounded instance refers to the bin packing problem in which no more than C items could be released for some constant C. In this section we will determine the competitive ratio of the best possible online algorithm for the bounded instances via a dynamic programming algorithm. Indeed, a best algorithm for bounded instances could also be simply determined by brute force. However, as it needs to be further generalized, the dynamic programming algorithm will provide additional information on its structure.

We establish a layered graph G, in which there are $|ST_h|$ vertices at the h-th layer, each corresponding to some η_i^h. With a slight abuse of the notation we also use η_i^h to denote its corresponding vertex. For every η_i^h, we construct B vertices, namely $\alpha_{i,j}^h$ for $1 \leq j \leq B$ representing the release of item of size j by the adversary. For simplicity, all the $\alpha_{i,j}^h$ are denoted as vertices of the $(h + 1/2)$-th layer. There are only edges between vertices of the h-th layer and the $(h + 1/2)$-th layer, and between vertices of the $(h + 1/2)$-th layer and the $(h + 1)$-st layer. Indeed, there is an edge between η_i^h and $\alpha_{i,j}^h$ for any h, i and $1 \leq j \leq B$. There is an edge between $\alpha_{i,j}^h$ and η_k^{h+1}, if by packing the item of size j into a certain bin, the state η_i^h is changed to η_k^{h+1}.

Now we can easily associate an online algorithm with a path in the layered graph G. If the adversary releases n items of sizes $J_1, J_2, \cdots, J_n$, and meanwhile the algorithm returns a series of states $\eta_{i_0}^0$ (obviously $i_0 = 1$ since ST_0 contains only one element), $\eta_{i_1}^1$, $\cdots$, $\eta_{i_n}^n$, then associate it with a path in the graph as $\eta_{i_0}^0 \to \alpha_{i_0,j_1}^0 \to \eta_{i_1}^1 \to \cdots \to \alpha_{i_{n-1},j_n}^{n-1} \to \eta_{i_n}^n$.

Meanwhile, any path of length $2n$ that starts at η_1^0 and ends at η_i^n for some i represents the packing of n items by a certain online algorithm. We adopt the idea of [4] to reformulate the problem of finding the best online algorithm for bounded instances into the following problem on a game between the adversary and the packer:

- Initially the game starts at the vertex η_1^0.
- If currently the game arrives at the vertex η_i^h for $h < C$, then the adversary either decides to end the game, or moves the game to some α_{ij}^h that is incident to η_i^h. If the game arrives at the vertex η_i^h for $h = C$, then the game ends.
- If currently the game arrives at the vertex α_{ij}^h, then the packer chooses to move the game to some η_k^{h+1} that is incident to α_{ij}^h.

Again we take the above figure as an example. Starting at η_1^0, if the adversary releases an item of size 1, he moves the game to $\alpha_{1,1}^0$. Then the packer packs this item into one bin, meaning that he moves the game to η_1^1. Now the adversary could either choose to stop the game, or continue to release items. If he releases a new item of size 1, the game arrives at $\alpha_{1,1}^1$.

If the game ends at η_i^h for $h \le C$, then the utility of the adversary is defined to be $\tilde{\rho}(\eta_i^h)$, while the utility of the packer is defined to be $-\tilde{\rho}(\eta_i^h)$. Starting from η_1^0, if the packer always packs items according to Next-Fit, then obviously the adversary could choose to release a certain list of items such that the game ends at some η_i^h with $\tilde{\rho}(\eta_i^h)$ about 2. If the packer is smart enough, he would resort to an optimum online algorithm (with the competitive ratio of ρ^*) so that no matter how the adversary releases items, he is always able to move the game to some η_i^h with $\tilde{\rho}(\eta_i^h) \le \rho^*$. Thus, $-\rho^*$ is the largest possible utility the packer could achieve starting at η_1^0, and meanwhile ρ^* is the largest possible utility the adversary could ever achieve. Analogously, we define $\rho(\eta_i^h)$ to be the largest utility the adversary could get by releasing at most $C - h$ additional items, which implies that starting at η_i^h, the optimum "online" algorithm would have a competitive ratio of $\rho(\eta_i^h)$. Now we provide a dynamic programming algorithm to compute the value of $\rho(\eta_i^h)$. Obviously we have $\rho(\eta_i^h) \ge \tilde{\rho}(\eta_i^h)$.

Note that the adversary is no longer able to release items if the current scenario is some $\eta_i^C \in ST_C$. Thus we have $\rho(\eta_i^C) = \tilde{\rho}(\eta_i^C)$.

Let $N(\alpha_{i,j}^h)$ be the set of vertices at the $(h+1)$-st layer that are incident to the vertex $\alpha_{i,j}^h$, for any i, j and $h \le C - 1$. Then:

$$\rho(\alpha_{i,j}^h) = \min_k \{\rho(\eta_k^{h+1}) | \eta_k^{h+1} \in N(\alpha_{i,j}^h)\}, \rho(\eta_i^h) = \max\{\tilde{\rho}(\eta_i^h), \max_j\{\rho(\alpha_{i,j}^h)\}\}.$$

We give an explanation for the first equation, and the second one is similar. Suppose currently the game is at η_i^h. The adversary knows that if he releases an item of size j, then all the possible states by packing this new item are given by $\{\eta_k^{h+1} \in N(\alpha_{i,j}^h)\}$. Suppose the adversary is clever enough, who knows that if the game arrives at η_k^{h+1}, the best possible online algorithm, starting at η_k^{h+1}, would have a competitive ratio of $\rho(\eta_k^{h+1})$. Thus, if he chooses to release an item of size j, the largest utility he could get is $\rho(\alpha_{i,j}^h) = \min_k\{\rho(\eta_k^{h+1}) | \eta_k^{h+1} \in N(\alpha_{i,j}^h)\}$ as the "worst case" for him is that the packer chooses to pack items in the way indicated by $\min_k\{\rho(\eta_k^{h+1}) | \eta_k^{h+1} \in N(\alpha_{i,j}^h)\}$.

A best possible online algorithm for bounded instances is described below.

Algorithm 1

1. For a given constant C, construct the graph G and calculate the $\tilde{\rho}(\eta_i^C)$.
2. For all i, let $\rho(\eta_i^C) = \tilde{\rho}(\eta_i^C)$.
3. For $h = C-1$ down to 1, iteratively calculate, for all i, j,

$$\rho(\alpha_{i,j}^h) = \min_k\{\rho(\eta_k^{h+1})|\eta_k^{h+1} \in N(\alpha_{i,j}^h)\},\ \rho(\eta_i^h) = \max\{\tilde{\rho}(\eta_i^h) \max_j\{\rho(\alpha_{i,j}^h)\}\}.$$

4. For the released item of size j when the current state is η_i^h, let $k^* = \text{argmin}_k$ $\{\rho(\eta_k^{h+1})|\eta_k^{h+1} \in N(\alpha_{i,j}^h)\}$, then we assign this item to the bin such that the state η_i^h is changed to the state $\eta_{k^*}^{h+1}$.

Remark. For simplicity we will call $\rho(\eta_i^h)$ as the *ratio* of the state η_i^h. Furthermore for the ease of analysis we will round up each instant ratio to be its nearest value in $SV = \{1, 1+\epsilon, 1+2\epsilon, \cdots, 2\}$, and as a consequence after computation the ratios of states also belong to SV.

4 From Modified Instances to General Instances

Let A be the best possible online algorithm for bounded instances. As we have mentioned, we need to generalize it to an algorithm for general instances, and the generalization has two steps. First, we generalize it to an algorithm for the modified instances (the definition of a modified instance will be given below). Then, we generalize the algorithm for modified instances to an algorithm for general instances. We deal with the easier part in this section, i.e., roughly speaking, we show that an algorithm of competitive ratio ρ^* for modified instances could be transformed into an algorithm of competitive ratio $\rho^* + O(\epsilon)$ for the general instances.

We give the definition of a modified instance. Let $l = (J_1, \cdots, J_h)$ be any list of h items ($|l| = h$). Given l, we use kl to denote the sequence by duplicating each item of l into k items, i.e., $kl = (J'_1, \cdots, J'_{kh})$, where $J'_{ki+j} = J_{i+1}$ for $0 \le i \le h-1$ and $1 \le j \le k$, or equivalently, $kl = (\underbrace{J_1, \ldots, J_1}_{k}, \ldots, \underbrace{J_j, \ldots, J_j}_{k})$.

Given any integers $k, c > 0$, we say L is a modified instance or a modified list (with respect to (k, c)), if $L = (l_1, kl_2, k^2 l_3, \cdots, k^h l_{h+1})$, where $|l_i| = c$ for $1 \le i \le h$, and $|l_{h+1}| \le c$. The bin packing problem for modified instances is the bin packing problem satisfying the following conditions:

- The items released by the adversary form a modified list.
- The adversary could only stop releasing items at certain times, i.e., he could only do the following:
  - The adversary releases no more than c items, and stops.
  - The adversary releases no more than $c+kc$ items, and stops after he releases the $(c+kj)$-th item ($j \le c$).
  - $\cdots$

- The adversary releases no more than $c + kc + \cdots + k^h c$ items, and stops after he releases the $(c + kc + \cdots + k^h j)$-th item $(j \leq c)$.

Theorem 2. *Given any $\epsilon > 0$, if there is an online algorithm of competitive ratio ρ^* on modified instances with respect to any $k > 0$ and $c \geq c(k, B, \epsilon)$ (where $c(k, B, \epsilon)$ is a constant only depending on k, B and ϵ), then there is an algorithm of competitive ratio $\rho^* + \epsilon$ for the general problem.*

Proof. We prove the theorem by modifying the algorithm A of competitive ratio ρ^* for modified instances. Throughout the proof we keep track of two lists, one is the list σ of items released in the general bin packing problem, and the other is the item list σ' of a modified instance which is constructed from σ so that algorithm A could return a feasible packing by taking σ' as an input.

In the following, we construct an algorithm for the general problem with the input σ. If the h-th item in σ arrives, and $h \leq c$, we pack this item according to algorithm A. We have $\sigma = \sigma'$. When the $(c+1)$-st item in σ, say, J_{c+1} releases, we take it as k identical items, one true item and $k-1$ fake items, and pack them according to A. Now we add k copies of J_{c+1} to σ'. Consider the $(c+2)$-nd item. If it is different from J_{c+1}, then again we take it as k identical J_{c+2} and pack them according to A. Meanwhile we add J_{c+2} to σ and k copies of J_{c+2} to σ'. Otherwise, it is the same with J_{c+1}, then we replace one fake J_{c+1} with this item in the current packing. In this case, σ' remains the same. We proceed with the above procedure. Whenever a new item J_n releases, we add it to σ and check if there exists a fake item of the same size in the current solution. If yes, we replace this fake item with this new item and σ' remains the same. Otherwise, we add J_n to σ', and another $k^h - 1$ identical items are released together with it for some h depending on the length of σ'. Now we resort to A to decide a packing for these k^h items, and for J_n, there are $k^h - 1$ fake items now.

Next we check the competitive ratio of the above algorithm. Let μ be the number of bins used, r_i be the number of items of size i according to σ, r_i' be the number of items of size i according to σ'. We use $|\sigma|$ to denote the number of items in the list σ and suppose $c + kc + \cdots + k^{h-1}c < |\sigma'| \leq c + kc + \cdots + k^h c$ for some h. Then it follows that $r_i' - r_i \leq k^h$ according to the construction of σ'.

Since the competitive ratio of A is ρ^*, we have $\mu \leq \rho^* OPT(r_1', \cdots, r_B') + \kappa$. Meanwhile, $OPT(r_1, \cdots, r_B) \geq OPT(r_1', \cdots, r_B') - Bk^h$ as $r_i' - r_i \leq k^h$. Notice that $OPT(r_1', \cdots, r_B') \geq |\sigma'|/B > k^{h-1}c/B$. The competitive ratio for the general bin packing is at most (for simplicity we let $OPT = OPT(r_1', \cdots, r_B')$).

$$\frac{\mu - \kappa}{OPT(r_1, \cdots, r_B)} \leq \frac{\rho^* OPT}{OPT - Bk^h} = \frac{\rho^*}{1 - Bk^h/OPT} \leq \frac{\rho^*}{1 - B^2k/c}.$$

The theorem follows by taking $c > B^2k/\epsilon^2 = c(k, B, \epsilon)$.

Remark. For ease of analysis, the following sequence is also taken to be a modified instance (with respect to (k, c)): $L = (l_1, kl_2, k^2 l_3, \cdots, k^h l_{h+1})$ where $|l_i| = c$ for $2 \leq i \leq h$, $|l_{h+1}| \leq c$ and $|l_1| \geq c$, i.e., the first part of the list could contain more than c items.

5 From Bounded Instances to Modified Instances

Let ρ_0 be the competitive ratio of the best algorithm for bounded instances (in which the adversary releases at most C items). We show in this section that when C is large enough, we can transform the algorithm into a $(\rho_0+O(\epsilon))$-competitive algorithm for the bounded instances with respect to (k_0, c_0) for some k_0 and c_0. Combining this result with Theorem 2, Theorem 1 follows directly. The values of C, k_0 and c_0 will be determined at the end of this section.

5.1 Overview of the Technique

We revisit the graph G that contains all the possible states. G is an infinite graph and we can only afford to compute the ratios of states in ST_h for $h \le C$. Note that once the adversary releases an i-th item with $i \le C$, the optimal algorithm for bounded instances can refer to the ratio of the current state to decide how to pack this item (the reader may refer to Algorithm 1 in Sect. 3). What if the current state is some η_i^n for $n > C$? A natural idea is to do *state mapping*, i.e., we map η_i^n to some proper $\eta_{i'}^h$ for $h < C$. Once a new item is released, we check $\eta_{i'}^h$ to see how this new item is packed, and then pack it in a similar way for η_i^n.

Modified instances are defined in order that we can carry out the above idea. Roughly speaking, we will specify some constants γ and k such that $k\gamma < C$, and take states of ST_h for $\gamma \le h \le k\gamma$ as samples. Consider modified instances with respect to $(k, (k-1)\gamma)$, i.e., the item lists of the form $(l_1, kl_2, \cdots, k^{h-1}l_h, k^h l_{h+1})$, where $|l_1| = k\gamma$, $|l_i| = (k-1)\gamma$ for $2 \le i \le h$ and $|l_{h+1}| \le (k-1)\gamma$.

Suppose the adversary releases at most $k\gamma$ items. Obviously we can run the algorithm for bounded instances as $k\gamma < C$. Otherwise, suppose $k\gamma$ items are released and the current state is some $\eta_i^{k\gamma} = (r, x)$. Then according to our definition of neighborhood, with some slight perturbation Δ_1 we have $\eta_i^{k\gamma} = k\eta_{i'}^{\gamma} + \Delta_1$, where $\eta_{i'}^{\gamma}$ is some state of ST_γ. By the definition of modified instances, after $k\gamma$ items the adversary releases k identical items, denoted as kJ_j for simplicity, where J_j is arbitrary. According to the algorithm, for bounded instances, $\eta_{i'}^{\gamma}$ changes to $\eta_{h'}^{\gamma+1}$, when a single item J_j is released. We write $\eta_{h'}^{\gamma+1} = \eta_{i'}^{\gamma} + J_j$ for simplicity. Then we can pack kJ_j in a way such that $k\eta_{i'}^{\gamma} + kJ_j = k\eta_{h'}^{\gamma+1}$, e.g., if a new bin is opened for J_j in $\eta_{i'}^{\gamma}$, then k new bins are opened for kJ_j in $k\eta_{h'}^{\gamma+1}$. Now we have $\eta_i^{k\gamma} + kJ_j = k\eta_{h'}^{\gamma+1} + \Delta_1$, and if the adversary releases the next k identical items, we check $\eta_{h'}^{\gamma+1}$ to see how to pack them.

After the adversary release $k\gamma + k(k-1)\gamma = k^2\gamma$ items, we arrive at some state of $ST_{k^2\gamma}$ which could be expressed as $k\eta_\ell^{k\gamma} + \Delta_1$ for some $\eta_\ell^{k\gamma} \in ST_{k\gamma}$, and again with some slight perturbation Δ_2 we have $\eta_\ell^{k\gamma} = k\eta_{\ell'}^{\gamma} + \Delta_2$. Hence the current state is $k^2\eta_{\ell'}^{\gamma} + k\Delta_2 + \Delta_1$. According to the definition of modified instances, next, the adversary releases k^2 identical items, say, k^2J_j. Now we can again check how a single item J_j is packed for $\eta_{\ell'}^{\gamma}$ and carry on the above arguments. To make the above arguments work, we need to show the following conditions: Given a state η^h, a slight perturbation (changing η^h to $\eta^h + \Delta$) does not change the instant ratio much; and multiplication by an integer k (changing

η^h to $k\eta^h$) does not change the instant ratio much. It results in the following two lemmas (the complete proofs will be given in a full version of the paper).

Lemma 1. *For any integers $k, d > 0$, and for any $(r(d), x(d))$ such that $\sum r_i(d) = d$, we have $kOPT(r(d)) - kB^B \leq OPT(kr(d)) \leq kOPT(r(d))$.*

Note that $OPT(r(d)) \geq d/B$. Take $d = B^{B+2}/\epsilon$ so that $B^B \leq \epsilon OPT(r(d))/B$. Recall that $R = ST_d$. Let $C = 2^\mu d$ for some constant μ. Then we can calculate the ratio of each state of ST_h for $h \leq C$, and an optimal algorithm for bounded instances could be determined. Let ρ_0 be its competitive ratio. Let $q \geq d$. Two states of the q-th layer, say, $\eta_i^q = (r(q), x(q))$ and $\eta_j^q = (r(q)', x(q)')$, are called *near*, if $|r_i(q) - r_i(q)'| \leq q/d$ and $|x_i(q) - x_i(q)'| \leq q/d$. We have

Lemma 2. *For any two near states $\eta_i^q = (r(q), x(q))$ and $\eta_j^q = (r(q)', x(q)')$, $|\rho(r(q), x(q)) - \rho(r(q)', x(q)')| \leq O(\epsilon)$.*

The above lemma implies that the ratio of a state in kR differs at most $O(\epsilon)$ to the ratio of its neighbors for any integer $k > 0$.

5.2 Constructing an Algorithm for Modified Instances

Recall that $R = ST_d$. For any integer $k > 0$, the states in kR are called *principle states* of ST_{kd}. We consider $ST_{2^h d}$ for $h = 1, 2, \cdots, \mu$. There is a vertex for each state of $2^h R$, and as we mention before, all the states of $ST_{2^h d}$ could be partitioned into subgroups where each group consists of a state in $2^h R$ and its neighbors. Since a state not in $2^h R$ might be a neighbor of multiple principle states, as discussed in Sect. 2, we give an assignment so that it becomes a neighbor of a unique principle state.

An assignment is called *compatible*, if according to this assignment, $(r', x') \in ST_{2^h d}$ is a neighbor of $(r, x) \in 2^h R$ implies that $(2^k r', 2^k x') \in ST_{2^{h+k} d}$ is a neighbor of $(2^k r, 2^k x) \in 2^{h+k} R$ for any $k \geq 1$. A compatible assignment could be constructed easily. In the following discussion we take one arbitrary compatible assignment. We use $T(r, x)$ to denote the set of neighbors of any $(r, x) \in kR$ (including (r, x)). Define

$$\rho(T(r, x)) = \min\{\rho(r', x') | (r', x') \in T(r, x)\}.$$

Since for any h the set $ST_{2^h d}$ is always partitioned into $|R|$ subgroups with each group being the set of neighbors of some principle state, we sort the states of $|R|$ in an arbitrary sequence as $\eta_1^d, \eta_2^d, \cdots, \eta_{|R|}^d$, where $\eta_i^d = (r(d), x(d))$ for some $r(d)$ and $x(d)$. We denote as $k\eta_i^d = (kr(d), kx(d))$.

Determining the Parameters. Consider $(\rho(T(2^h\eta_1^d)), \rho(T(2^h\eta_2^d)), \cdots, \rho(T(2^h\eta_{|R|}^d)))$ for $h = h_0, h_0 + 1, \cdots, \mu$, where h_0 is some constant. Each coordinate takes some value of $1 + k\epsilon$ for $0 \leq k \leq 1/\epsilon$ and thus has at most $1/\epsilon + 1$ different possible values. Each vector contains $|R|$ coordinates. There are at most $(1 + 1/\epsilon)^{|R|}$ different vectors. Thus, let $\mu - h_0 = (1 + 1/\epsilon)^{|R|}$. Among the $\mu - h_0 + 1$ vectors, we know that there exist two vectors which are identical. Let them be

$(\rho(T(\xi\eta_1^d)), \cdots, \rho(T(\xi\eta_{|R|}^d)))$ and $(\rho(T(\lambda\xi\eta_1^d)), \cdots, \rho(T(\lambda\xi\eta_{|R|}^d)))$ for some integers λ and ξ. Then $\lambda \le 2^{\mu-h_0} \le 2^{(1+\epsilon)^{|R|}}$, which is a constant that depends on $|R|$ and $1/\epsilon$, i.e., B and $1/\epsilon$.

According to the above arguments, we can first apply Theorem 2 with $k = 2^{(1+\epsilon)^{|R|}}$ and determine the parameter $c(k, B, \epsilon)$ that only depends on k, B and ϵ in the theorem. Let it be c_0. Then we take h_0 such that $2^{h_0 d} \ge c_0$ and let $\mu = h_0 + (1 + 1/\epsilon)^{|R|}$. Now we take $C = 2^\mu d$ (recall that $d = B^{B+2}/\epsilon$) and compute the ratios of each state and find the two identical vectors from $(T(2^h\eta_1^d), \cdots, T(2^h\eta_{|R|}^d))$ for $h = h_0, h_0+1, \cdots, \mu$. Again we denote the two identical vectors we find out as $(\rho(T(\xi\eta_1^d)), \cdots, \rho(T(\xi\eta_{|R|}^d)))$ and $(\rho(T(\lambda\xi\eta_1^d)), \cdots, \rho(T(\lambda\xi\eta_{|R|}^d)))$.

Let ρ_0 be competitive ratio of the best possible algorithm for bounded instances in which the adversary releases at most C items.

Theorem 3. *There exists an online algorithm whose competitive ratio is* $\rho_0 + O(\epsilon)$ *for the modified instance with respect to* (k, c)*, where* $k = \lambda$ *and* $c = (\lambda - 1)\xi d$*.*

Proof. Before starting the proof, recall that in order to apply Theorem 2, we need to show that $c = (\lambda - 1)\xi d \ge c(k, B, \epsilon) = c(\lambda, B, \epsilon)$. Since $\lambda \le 2^{(1+\epsilon)^{|R|}}$, we have $c(\lambda, B, \epsilon) \le c(2^{(1+\epsilon)^{|R|}}, B, \epsilon) = c_0$. Meanwhile, $2^{h_0 d} \ge c_0$, thus $(\lambda - 1)\xi d \ge \xi d \ge 2^{h_0} d \ge c_0 \ge c(\lambda, B, \epsilon)$.

Now we come to the proof of the theorem. Let A be an optimal algorithm for the bounded instance. Now the list of items released by the adversary is of the form $l = (l_1, \lambda l_2, \lambda^2 l_3, \cdots, \lambda^h l_{h+1})$ where $|l_1| = \lambda\xi d$, $|l_i| = (\lambda - 1)\xi d$ for $2 \le i \le h$ and $|l_{h+1}| \le (\lambda - 1)\xi d$. Obviously we can always apply A for the first $\lambda\xi d$ items in the list of items released by the adversary. It remains to show how to pack the list λl_2.

For any $T(\xi\eta_i^d)$, there exists some $(r, x) \in T(\xi\eta_i^d)$ such that $\rho(r, x) = \rho(T(\xi\eta_i^d))$. Denote it as $T_{min}(\xi\eta_i^d)$. Suppose after packing the c items with A, the current state is some state, say, $z_1 \in T(\lambda\xi\eta_h^d)$ for some h. Then

$$\rho(T(\lambda\xi\eta_h^d)) \le \rho(z_1) \le \rho_0.$$

Based on the selection of ξ and λ, we have

$$\rho(T_{min}(\xi\eta_i^d)) = \rho(T(\xi\eta_h^d)) = \rho(T(\lambda\xi\eta_h^d)) \le \rho_0.$$

According to the compatible assignment, $\lambda T_{min}(\xi\eta_i^d) \in T(\lambda\xi\eta_i^d)$, and is near z_1. Let $z_1^* = \lambda T_{min}(\xi\eta_i^d)$, and we let $z_1 = z_1^* + \Delta_1$, where $||\Delta_1||_\infty \le \lambda\xi$, i.e., the absolute value of each coordinate of Δ_1 is bounded by $\lambda\xi$. Suppose the first λ items of the list are λJ_j. According to A, $z_1^*/\lambda \in T_{min}(\xi\eta_i^d)$ changes to y by adding a single item J_j. Now if we start at the state z_1^* to pack λJ_j, we may view z_1^* as λ copies of z_1^*/λ and we can pack the λ identical items in the same way, i.e., pack items in the way that z_1^* changes to λy. Thus, starting at z_1 to pack these items, we may adopt the same idea as in the proof of Lemma 2. Again, add some dummy bins to alter the state into z_1^* and pack items. It follows that z_1 changes to $\lambda y + \Delta_1$.

We pack items iteratively as the above procedure. Let $z_2 = z_1^*/\lambda + l_2$ according to Algorithm A. Then it can be easily seen that $z_2 \in ST_{\lambda\xi d}$. Meanwhile, the above way of packing cause the current state to be $\lambda z_2 + \Delta_1$. Recall that the algorithm A ensures that $\rho(z_2) \le \rho(z_1^*/\lambda) \le \rho_0$. Suppose $z_2 \in T(\lambda\xi\eta_{h'}^d)$ for some h'. Again we get

$$\rho(T(\xi\eta_{h'}^d)) = \rho(T(\lambda\xi\eta_{h'}^d)) \le \rho_0.$$

Thus there exists some $z_2^* \in T(\lambda\xi\eta_{h'}^d)$ such that $\rho(z_2^*/\lambda) \le \rho_0$. Again z_2 is near z_2^* and we have $z_2 = z_2^* + \Delta_2$ for some $||\Delta_2||_\infty \le \lambda\xi$. Suppose $z_2^*/\lambda + l_3 = z_3$ according to Algorithm A. Starting at $\lambda z_2 + \Delta_1$, the next part of the list is $\lambda^2 l_3$. Thus $\lambda z_2 + \Delta_1 + \lambda^2 l_3 = \lambda^2 z_3 + \lambda\Delta_2 + \Delta_1$.

Iteratively applying the above computation, the final state arrived is $\lambda^h z_{h+1} + \lambda^{h-1}\Delta_h + \cdots + \Delta_1$, where $||\Delta_i||_\infty \le \lambda\xi$, $\rho(z_{h+1}) \le \rho_0$. Due to the final part of the list $\lambda^h l_{h+1}$ that may not be complete, i.e., $|l_{h+1}|$ may not equal to c, z_{h+1} may not be a state of $ST_{\lambda\xi d}$. Nevertheless, z_{h+1} is some state between the $\lambda\xi d$-th layer and ξd-th layer and Algorithm A ensures that $\rho(z_{h+1}) \le \rho_0$.

Compute the instant ratio of $\lambda^h z_{h+1} + \lambda^{h-1}\Delta_h + \cdots + \Delta_1$. Let $z_{h+1} = (r, x)$. Recall Lemma 1. Then $OPT(\lambda^h r) \ge \lambda^h OPT(r) - \lambda^h B^B$.

$$\begin{aligned}\tilde{\rho}(\lambda^h z_{h+1} + \lambda^{h-1}\Delta_h + \cdots + \Delta_1) &\le \frac{\lambda^h \sum x_i + B\lambda\xi \sum_{j=1}^h \lambda^{j-1}}{OPT(\lambda^h r) - B\lambda\xi \sum_{j=1}^h \lambda^{j-1}} \\ &\le \frac{\lambda^h \rho_0 OPT(r) + B\lambda\xi \cdot 2\lambda^{h-1}}{\lambda^h OPT(r) - \lambda^h B^B - B\lambda\xi \cdot 2\lambda^{h-1}} \\ &= \frac{\rho_0 OPT(r) + 2B\xi}{OPT(r) - B^B - 2B\xi}\end{aligned}$$

Since $z_{h+1} = (r, x)$ is a state between the $\lambda\xi d$-th layer and ξd-th layer, we know that $OPT(r) \ge \xi d/B$. As $d = B^{B+2}/\epsilon$, it follows directly that $\tilde{\rho}(\lambda^h z_{h+1} + \lambda^{h-1}\Delta_h + \cdots + \Delta_1) \le \rho_0 + O(\epsilon)$.

6 Concluding Remarks

In this paper we have designed a competitive scheme for online bin packing such that the competitive ratio of our algorithm is at most of $1 + \epsilon$ times the best possible competitive ratio of any online algorithms, for any given $\epsilon > 0$. Our scheme provided a theoretical approach to narrow the known lower bound 1.54037 [1] and the upper bound 1.58889 [14]. The running time of our scheme is exponential in the bin size B and $1/\epsilon$. If the number of item sizes is a constant, our algorithm runs in polynomial time. But it remains an open problem whether we can design competitive schemes polynomially in both the number of items and $\log B$.

For bin packing, the absolute competitive ratio is another measure for online algorithms in the literature, though it is not as common as the asymptotic competitive ratio. To the knowledge of us, the best known lower bound is 5/3 [3] and the best known upper bound is 1.7 [5] in terms of absolute competitive ratio. Note that the results in this work are also valid even if the performance metric is the absolute competitive ratio. In addition, we claim that the techniques used in this paper can be extended to other variants of bin packing problems, such as the online variable-sized bin packing problem and the online bounded-space bin packing problem.

References

1. Balogh, J., Békési, J., Galambos, G.: New lower bounds for certain classes of bin packing algorithms. In: Jansen, K., Solis-Oba, R. (eds.) WAOA 2010. LNCS, vol. 6534, pp. 25–36. Springer, Heidelberg (2011)
2. Boyar, J., Epstein, L., Favrholdt, L.M., et al.: The maximum resource bin packing problem. Theoret. Comput. Sci. **362**, 127–139 (2006)
3. Brown, D., Baker, B.S., Katseff, H.P.: Lower bounds for on-line two-dimensional packing algorithms. Acta Informatica **18**, 207–225 (1982)
4. Chen, L., Ye, D., Zhang, G.: Approximating the optimal competitive ratio for an ancient online scheduling problem. CoRR, abs/1302.3946v1 (2013)
5. Dośa, G., Sgall, J.: First fit bin packing: A tight analysis. In: Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS), pp. 538–549 (2013)
6. Fernandez de La Vega, W., Lueker, G.S.: Bin packing can be solved within 1+ ε in linear time. Combinatorica **1**, 349–355 (1981)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the theory of of NP-Completeness. Freeman and Company, San Francisco (1979)
8. Günther, E., Maurer, O., Megow, N., Wiese, A.: A new approach to online scheduling: Approximating the optimal competitive ratio. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 118–128 (2013)
9. Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L.: Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J. Comput. **3**, 256–278 (1974)
10. Johnson, D.S.: Fast algorithms for bin packing. J. Comput. Syst. Sci. **8**, 272–314 (1974)
11. Karmarkar, N., Karp, R.: An efficient approximation scheme for the one-dimensional bin-packing problem. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS), pp. 312–320 (1982)
12. Lee, C.C., Lee, D.T.: A simple online bin packing algorithm. J. ACM **32**, 562–572 (1985)
13. Ramanan, P., Brown, D., Lee, C.C., Lee, D.T.: Online bin packing in linear time. J. Algorithms **10**, 305–326 (1989)
14. Seiden, S.: On the online bin packing problem. J. ACM **49**, 640–671 (2002)
15. Ullman, J.: The performance of a memory allocation algorithm. Technical report. Princeton University. Dept. of Electrical Engineering (1971)
16. Yao, A.C.C.: New algorithms for bin packing. J. ACM **27**, 207–227 (1980)
17. Y. Zhang, F.Y.L. Chin, H.-F. Ting et al. Online algorithms for 1-space bounded 2-dimensional bin packing and square packing. Theortical Comput. Sci. (2014) (to appear)

Springer
http://www.springer.com/978-3-319-12690-6

Combinatorial Optimization and Applications
8th International Conference, COCOA 2014, Wailea, Maui, HI, USA, December 19-21, 2014, Proceedings
Zhang, Z.; Wu, L.; Xu, W.; Du, D.-Z. (Eds.)
2014, XV, 774 p. 218 illus., Softcover
ISBN: 978-3-319-12690-6