

# A Fast Asymptotic Approximation Scheme for Bin Packing with Rejection

Wolfgang Bein<sup>1\*</sup> José R. Correa<sup>2\*\*</sup> Xin Han<sup>3\*\*\*</sup>

<sup>1</sup> Center for the Advanced Study of Algorithms, School of Computer Science,  
University of Nevada, Las Vegas, NV 89154, USA,

`bein@cs.unlv.edu`,

<sup>2</sup> School of Business, Universidad Adolfo Ibáñez, Santiago, Chile,

`correa@uai.cl`

<sup>3</sup> School of Informatics, Kyoto University, Kyoto 606-8501, Japan,  
`hanxin@kuis.kyoto-u.ac.jp`

**Abstract.** The *bin packing with rejection* problem is the following: Given a list of items with associated sizes and rejection costs, find a packing into unit bins of a subset of the list, such that the number of bins used plus the sum of rejection costs of unpacked items is minimized. In this paper, we first show that bin packing with rejection can be reduced to  $n$  multiple knapsack problems. Then, based on techniques for the multiple knapsack problem we give a fast asymptotic polynomial time approximation scheme with time complexity  $O(n^{O(\epsilon^{-2})})$ . This improves a recent approximation scheme given by Epstein, which has time complexity  $O(n^{O((\epsilon^{-4})^{\epsilon^{-1}})})$ . Finally, we show that our algorithm can be extended to variable-sized bin packing with rejection and give an asymptotic polynomial time approximation scheme for it.

## 1 Introduction

In the bin packing problem items of specified size have to be packed into the minimum number of unit bins. This problem, of particular interest in operations research and computer science, has been extensively studied since the Sixties (see [3] for a detailed survey). Interestingly, although bin packing is NP-hard, efficient algorithms for computing solutions which are arbitrarily close to optimal (asymptotically) were obtained early on by Fernandez de la Vega and Luecker [6], and by Karmarkar and Karp [9]. More recently, packing and scheduling problems with rejection penalties, which are natural variants of their classic counterparts, have received attention (see e.g. [1, 4, 8]). Here one can decide to refuse an item, but in doing so one is charged a certain cost. This setting is of interest since in many practical situations one may choose to leave some items unpacked (or unscheduled) at a certain price. For instance, this problem arises naturally in

---

\* Research conducted while visiting Japan as Kyoto University Visiting Professor.

\*\* Supported in part by CONICYT through grants FONDECYT 10600035 and ACT08.

\*\*\* Supported by NSFC (10231060).

the context of outsourcing tasks (e.g. shipments), in which a third party can take care of processing a task as long as they are paid for it. Other examples are in caching and data storage across a network, where an item can be cached or stored locally in advance, or, instead, may be fetched at cost when needed.

Formally, in the bin packing with rejection problem we are given a list  $I$  of items. Each item  $i \in I$  has an associated size  $s_i$ , and a rejection cost  $r_i$ . The goal is to find a partition of  $I$  into a set  $A$  of “accepted” items and a set  $R$  of “rejected” items, together with a packing of  $A$  into unit bins, such that the number of bins needed to pack  $A$  plus the total price of items in  $R$  is minimized. (Clearly the NP-hardness of the bin packing with rejection problem follows from that of bin packing.) This particular problem, was recently studied by He and Dósa [4], who obtain several online and off line results. In particular, they give an algorithm with asymptotic approximation ratio of  $3/2$ . Epstein [5] gave the first asymptotic polynomial time approximation scheme (APTAS) for the bin packing with rejection problem based on classical bin packing techniques.

Throughout this paper we will make use of the notion of *guessing*, see e.g. [2]. One “guesses” a quantity if one can construct a polynomial size set of values, which contains the desired value and a certificate can be constructed in polynomial time. By an abuse of terminology we will also use the term “guessing” for constructing the certificate itself.

**Our contribution.** Borrowing techniques for the multiple knapsack problem [2, 10], we provide an APTAS for bin packing with rejection which turns out to be more efficient than that of Epstein. Furthermore we extend our scheme to variable-sized bin packing with rejection and give an APTAS for this problem as well. Specifically, we show that the bin packing with rejection problem can be reduced to  $n$  multiple knapsack problems. Based on the techniques for the multiple knapsack problem, we give a fast asymptotic polynomial time approximation scheme with time complexity  $O(n^{O(\epsilon^{-2})})$  thereby improving the current best  $O(n^{O((\epsilon^{-4})^{\epsilon^{-1}})})$  scheme [5]. Our techniques directly apply to the variable-sized case.

The outline of this approach is as follows: i) First, we guess a packing cost  $cost_p$  and a rejection cost  $cost_r$  such that the two values are not much larger than those of an optimal solution, respectively. ii) Then, based on the two guessed values, we guess a sublist  $L_r$  of the input list  $L$  such that the total profit (rejection cost) in  $L_r$  is bounded by  $(1 + O(\epsilon)cost_r)$  and all items in  $L - L_r$  can be packed into bins with  $cost_p$ . iii) Finally, we call a bin packing algorithm to pack all the items in  $L - L_r$ . The key point is that we can guarantee that all the guesses above can be done in polynomial time of  $n$ , where  $n$  is the size of the input list.

## 2 An APTAS for the bin packing with rejection

We first show that for solving the problem of bin packing with rejection we only need to solve at most  $n$  multiple knapsack problems, where  $n$  is the size of the

input list. Note that any optimal value  $OPT(L)$  can be written as follows:

$$OPT(L) = OPT_p(L) + OPT_r(L),$$

where  $L$  is the input list,  $OPT_p(L)$  is the cost for packing, i.e., the number of bins used,  $OPT_r(L)$  is the total rejection cost for rejection.

**Definition 1.** Given a list  $L$ , we denote the total rejection cost in it by  $p(L)$ .

**Lemma 1.** Assume  $OPT_p$  is fixed, then the original problem is equivalent to the multiple knapsack problem of packing the input list into  $OPT_p$  bins to maximize the total profit of the bins.

*Proof.* Let  $OPT_p = i$  and  $L = L_p \cup L_r$ , where  $L_p, L_r$  are the packed list and the rejected list, respectively. Then we have

$$\begin{aligned} OPT(L) &= OPT_p(L) + OPT_r(L) \\ &= i + \sum_{j \in L_r} r_j \\ &= (i + \sum_{j \in L} r_j) - \sum_{j \in L_p} r_j \end{aligned}$$

By the above equation, if  $\sum_{j \in L_p} r_j$  is maximized, then  $OPT(L)$  reaches its minimum. In fact, maximizing  $\sum_{j \in L_p} r_j$  is equivalent to selecting a sublist  $L_p$  from the input list  $L$  and packing them into  $i$  bins to maximize the total rejection cost of the bins. Thus, if the knapsack problem of packing  $L$  into  $i$  bins is solved, then the original problem is solved as well.  $\square$

Lemma 1 naturally suggests the following algorithm:

- Guess  $OPT_p$  from 1 to  $n$ , i.e., the number of bins to be used.
- Consider rejection costs as profits, then call PTAS for the multiple knapsack problem [2, 10] to pack items into  $OPT_p$  bins and regard the unpacked items as the rejected items.
- Output  $\min\{OPT_p + R\}$  over all  $OPT_p$ , where  $R$  is the total cost of all rejected items.

Unfortunately, this does not yield an APTAS. In the above algorithm, the packing cost is  $OPT_p$ , i.e., the number of bins used, is  $OPT_p$ . Assume now that the optimal value of maximizing the total profit (rejection cost) packed into bins with  $OPT_p$  is  $X$  and the total profit packed by the PTAS of [2, 10] is  $(1 - \epsilon)X$ . Then the total cost from our algorithm is

$$\epsilon X + OPT(L).$$

The value of  $\epsilon X$  can be large, even larger than  $OPT(L)$ , thus we can not guarantee that the above algorithm always produces a solution near the optimal cost.

Instead, we focus on the rejected list to obtain our APTAS. More precisely, we guess a rejected list such that its total rejection cost is at most  $(1 + O(\epsilon))OPT_r(L)$  and such that the remaining items in  $L$  can be packed into bins with  $OPT_p(L)$ . Before we give details we recall the following lemma:

**Lemma 2.** [9] *For the bin packing problem there is an algorithm which runs in time polynomial in  $n$  and  $1/\epsilon$  and finds a solution that uses at most*

$$(1 + \epsilon)OPT + O(\epsilon^{-2})$$

*bins, where  $OPT$  is the optimal number of bins and  $\epsilon > 0$  is sufficiently small.*

Let  $OPT_p$  and  $OPT_r$  be the packing cost and the rejection cost in some optimal solution. There are three steps in our algorithm. First, we guess  $OPT_p$  and  $OPT_r$ . Then, based on these two values, we guess the packed list and the rejected list. Finally, we pack the packed list by a classical bin packing algorithm and reject all other items.

**Our algorithm.**

1. Guess the packing cost  $cost_p$  and the rejection cost  $cost_r$  such that  $cost_p = OPT_p$  and  $OPT_r \leq cost_r \leq (1 + \epsilon)OPT_r + 1$ .
2. Guess a rejected list  $L'_r$  such that  $cost_r \leq p(L'_r) \leq (1 + O(\epsilon))cost_r$ , where  $p(L'_r)$  is the total rejection cost in  $L'_r$ , and the remaining items  $L - L'_r$  can be packed  $cost_p$  bins.
3. Call APTAS to pack  $L - L'_r$  and reject all items in  $L'_r$  output the bins used and  $p(L'_r)$ .

**2.1 Guessing the rejection cost and the packing cost**

Since every item can be packed into one bin, the optimal value for  $n$  items is at most  $n$  and the optimal values for  $OPT_p$  and  $OPT_r$  are at most  $n$  as well. Then there are  $n + 1$  possibilities for  $OPT_p$ . For the value of the rejection cost  $OPT_r$ , if  $OPT_r \leq 1$  holds then we can instead estimate  $cost_r = 1$  for the asymptotic approximation. Else, if  $(1 + \epsilon)^i \leq OPT_r < (1 + \epsilon)^{i+1}$  holds then we have  $cost_r = (1 + \epsilon)^{i+1}$ , i.e., we guess the optimal rejection cost as  $(1 + \epsilon)^{i+1}$ , where  $\epsilon$  is the error bound and  $i \geq 0$  is an integer. Since  $OPT_r \leq OPT(L) \leq n$ , we have

$$i \leq \frac{\ln n}{\ln(1 + \epsilon)} \leq \frac{2 \ln n}{\epsilon},$$

where the last inequality follows from  $\ln(1 + \epsilon) \geq \epsilon - \epsilon^2/2$  and  $\epsilon \leq 1/2$ . Thus we have the following lemma:

**Lemma 3.** *We can guess  $cost_r$  and  $cost_p$  in time  $O(\epsilon^{-1}n \ln n)$  such that*

$$cost_p = OPT_p, \quad OPT_r \leq cost_r \leq (1 + \epsilon)OPT_r + 1,$$

*where  $\epsilon > 0$  is the error bound and  $OPT_p$  ( $OPT_r$ ) is the packed (rejection) cost in some optimal solution.*

## 2.2 Guessing the rejected list and the packed list

We briefly note the following simple lemma:

**Lemma 4.** *In a given input list  $L$ , for any item  $i \in L$ , if  $r_i > 1$  then in some optimal solution item  $i$  should be in the packed list.*

*Proof.* If an item  $i$  with  $r_i > 1$  is rejected in some optimal solution then we can reduce the total cost of the optimal solution by packing item  $i$  itself into a single bin.  $\square$

We describe how to guess the rejected items and the packed items such that the total cost of the rejected and packed items are near the optimal values, based on the rejection cost  $cost_r$  and the packed cost  $cost_p$  guessed in step 1. In the following, we denote the guessed rejected (packed) list by  $L'_r$  ( $L'_p$ ).

**How to guess the rejected list and the packed list.**

- 2.1 Place every item with rejection cost larger than 1 into packed list  $L'_p$  and all items with rejection cost smaller than  $1/n$  into the rejected list  $L'_r$ .
- 2.2 Consider the remaining items with the rejection cost in  $[1/n, 1]$ . There are three phases in assigning all the remaining items into  $L'_p$  and  $L'_r$ : i) rounding down each item's rejection cost to obtain  $O(\epsilon^{-1} \ln n)$  sublists such that in each sublist all the items have the same rejection cost, ii) then guessing the rejected items in each sublist and placing them into  $L'_r$  such that  $L'_r$  has its rejection cost in  $[cost_r, (1 + O(\epsilon)cost_r)]$ , iii) lastly, placing all remaining items into  $L'_p$ , testing the feasibility of packing  $L'_p$  into bins with  $cost_p$ .

Below are the details of the three phases in step 2.2; we refer to them as “rounding down”, “guessing the rejected list” and “testing the packed list.”

**Rounding down.** Given an item with a rejection cost  $r$ , we round  $r$  down to  $\bar{r}$  for some integer  $i \geq 0$  such that

$$\bar{r} = \frac{(1 + \epsilon)^i}{n} \leq r < \frac{(1 + \epsilon)^{i+1}}{n},$$

where  $\epsilon$  is the given error bound. Since  $r \leq 1$  we have  $\frac{(1+\epsilon)^i}{n} \leq 1$  and

$$i \leq \frac{\ln n}{\ln(1 + \epsilon)} \leq 2\epsilon^{-1} \ln n.$$

Then we get  $h \leq 2\epsilon^{-1} \ln n$  sublists  $L_i$  and in  $L_i$  all items have the same rejection cost  $\frac{(1+\epsilon)^i}{n}$ .

**Guessing the rejected list.** Let  $U$  be the rejected items in some optimal solution of  $\cup_i L_i$  and let  $U_i = L_i \cap U$ . Next, we enumerate a polynomial set of candidates for the rejected list such that one of them includes  $U$  and the total rejection cost in them is at most  $(1 + O(\epsilon))cost_r$ . Let  $p(U_i)$  be the total rejection cost in  $U_i$ . We select items from  $L_i$  as follows. We guess the values

$p(U_i)$  approximately for  $0 \leq i \leq h$ . For each  $i$  we guess a value  $k_i \in [0..h/\epsilon]$  such that

$$k_i(\epsilon \cdot \text{cost}_r/h) \leq p(U_i) \leq (k_i + 1)(\epsilon \cdot \text{cost}_r/h).$$

Then we guess the rejected items in each sublist  $L_i$ . Let  $a_i$  be the rejection cost of one item in  $L_i$ . For each  $k_i$ , for  $0 \leq i \leq h$ , we order all items in  $L_i$  in order of non-decreasing size values. Then if  $a_i > \epsilon \cdot \text{cost}_r/h$  then pick the largest  $\lceil k_i(\epsilon \cdot \text{cost}_r/h)/a_i \rceil$  items from  $L_i$  and put them into  $L'_r$ , else pick the largest  $\lfloor (k_i + 1)(\epsilon \cdot \text{cost}_r/h)/a_i \rfloor$  items from  $L_i$  and put them into  $L'_r$ . Later, we show all the candidates for  $k_i$  can be bounded by  $O(n^{\epsilon^{-2}})$ , i.e., there are a polynomial number of rejected lists.

**Testing the packed list.** Each time, after selecting the rejected item from  $L_i$ , we put the remaining items in  $L_i$  into the packed list  $L'_p$ . Then we try to use APTAS [9] to pack the items in  $L'_p$  into  $(1 + \epsilon)\text{cost}_p + O(\epsilon^{-2})$  bins. If all the items in  $L'_p$  can be packed then return  $L'_p$  and  $L'_r$ . Else we try another tuple  $(k_1, \dots, k_h)$ .

### 2.3 Analysis

Now we prove that the number of all candidate  $h$ -tuples  $(k_1, \dots, k_h)$  can be bounded by  $O(n^{O(1/\epsilon^2)})$ , which is polynomial in  $n$ . To this end, we first quote an important lemma obtained by Chekuri and Khanna [2]. (We include the short proof for self-containedness.)

**Lemma 5.** *Let  $f$  be the number of  $g$ -tuples of non-negative integers such that the sum of tuple coordinates is equal to  $d$ . Then  $f = \binom{d+g-1}{g-1}$ . If  $d + g \leq \alpha g$  then  $f = O(e^{\alpha g})$ .*

*Proof.* The first part of the lemma is elementary counting. If  $d + g \leq \alpha g$  then

$$f \leq \binom{\alpha g}{g-1} \leq \frac{(\alpha g)^{g-1}}{(g-1)!}.$$

Using Stirling's formula we can approximate  $(g-1)!$  by  $\sqrt{2\pi(g-1)}((g-1)/e)^{g-1}$ . Thus  $f = O((e\alpha)^{g-1}) = O(e^{\alpha g})$ .  $\square$

A naive way of guessing the values  $k_1, \dots, k_h$  requires  $n^{O(\ln n/\epsilon^2)}$  which is exponential. However, not all the values  $k_1, \dots, k_h$  are independent. Indeed, we have that

$$\sum_i \left( k_i \cdot \frac{\epsilon \cdot \text{cost}_r}{h} \right) \leq \sum_i p(U_i) = p(U) \leq \text{cost}_r,$$

where the last inequality follows from the inequality  $p(U) \leq OPT_r \leq \text{cost}_r$ . By the above observation, we have  $\sum_i k_i \leq h/\epsilon = O(\ln n/\epsilon^2)$ , then we get the following lemma.

**Lemma 6.** *Let an integer  $h \leq 2\epsilon^{-1} \ln n$ . Then the number of  $h$ -tuples  $(k_1, \dots, k_h)$  such that  $\sum_i k_i \leq h/\epsilon$  is  $O(n^{O(\epsilon^{-2})})$ .*

*Proof.* We apply the bound from Lemma 5, we have  $\alpha = (1 + 1/\epsilon)$  and  $g = 2\epsilon^{-1} \ln n$ , hence we get an upper bound  $e^{2\epsilon^{-1}(1+1/\epsilon) \ln n}$ .  $\square$

We introduce the notion of a “*small-packed*” solution:

**Definition 2 (Small-packed).** : *Given a solution of the packed list  $L_p$  and the rejected list  $L_r$ , if there are two items  $i \in L_p$  and  $j \in L_r$  such that*

$$r_i = r_j, \quad s_i \leq s_j,$$

*then we say the solution is small-packed, otherwise not small-packed.*

We can see if a solution is not *small-packed*, then there exist a packed item  $i$  and a rejected item  $j$  such that  $r_i = r_j$ ,  $s_i > s_j$ . Then we exchange the two items  $i$  and  $j$ , i.e., pack  $j$  and reject  $i$ , to get another solution without enlarging the total cost. By the same approach, we can prove the following lemma.

**Lemma 7.** *Any optimal solution can be reduced to a small-packed solution without changing the cost.*

To prove that our scheme works we also need the following lemma:

**Lemma 8.** *Assume all items have rejection costs equal to  $\frac{(1+\epsilon)^i}{n}$  for some integer  $i$ , where  $0 \leq i \leq h = O(\epsilon^{-1} \ln n)$ , and  $p(U) \leq cost_r \leq (1 + O(\epsilon))p(U) + 1$ , where  $U$  is a set of the rejected items in some optimal solution with the small-packed property. Then there exists a valid tuple  $(k_1, \dots, k_h)$  associated with  $L_r'$  such that  $U \subseteq L_r'$  and  $p(L_r') \leq (1 + O(\epsilon))p(U) + 1$ .*

*Proof.*  $U$  is a set of the rejected items in some optimal solution with the *small-packed* property. Then for all  $i$ , we define

$$k_i = \lfloor p(U_i)h / (\epsilon \cdot cost_r) \rfloor.$$

There are two cases based on the value of  $a_i$ , which is the rejection cost of one item in  $L_i$ . Assume there are  $x_i$  items in  $U_i$ . Then  $a_i \cdot x_i = p(U_i)$ .

- (i)  $a_i > \epsilon \cdot cost_r / h$ . We prove our selection from  $L_i$  is as the same as  $U_i = U \cap L_i$ , i.e.,  $U_i = L_i \cap L_r'$ . By the definition  $k_i = \lfloor p(U_i)h / (\epsilon \cdot cost_r) \rfloor$ , we have

$$\frac{p(U_i)h}{\epsilon \cdot cost_r} - 1 < k_i \leq \frac{p(U_i)h}{\epsilon \cdot cost_r}.$$

Then

$$p(U_i) - \frac{\epsilon \cdot cost_r}{h} < \frac{k_i \cdot \epsilon \cdot cost_r}{h} \leq p(U_i)$$

Since  $a_i > \epsilon \cdot cost_r / h$  and  $a_i \cdot x_i = p(U_i)$ ,

$$x_i - 1 < \frac{p(U_i)}{a_i} - \frac{\epsilon \cdot cost_r}{ha_i} < \frac{k_i \cdot \epsilon \cdot cost_r}{ha_i} \leq \frac{p(U_i)}{a_i} = x_i.$$

So we select  $\lceil (k_i \cdot \epsilon \cdot cost_r) / (ha_i) \rceil (= x_i)$  items from  $L_i$ . Since our selection is from large to small and the optimal solution with the rejected list  $U$  is *small-packed*, our selection from  $L_i$  is as the same as  $U_i = U \cap L_i$ , i.e.,  $U_i = L_i \cap L_r'$ .

- (ii)  $a_i \leq \epsilon \cdot \text{cost}_r/h$ . Here we prove that  $U_i \subseteq (L_i \cap L'_r)$  and  $p(L_i \cap L'_r) \leq p(U_i) + \frac{\epsilon \cdot \text{cost}_r}{h}$ . Again, by the definition  $k_i = \lfloor p(U_i)h/(\epsilon \cdot \text{cost}_r) \rfloor$ , we have

$$\frac{p(U_i)h}{\epsilon \cdot \text{cost}_r} < k_i + 1 \leq \frac{p(U_i)h}{\epsilon \cdot \text{cost}_r} + 1.$$

Then

$$p(U_i) < \frac{(k_i + 1) \cdot \epsilon \cdot \text{cost}_r}{h} \leq p(U_i) + \frac{\epsilon \cdot \text{cost}_r}{h}$$

So we select  $\lfloor ((k_i + 1)\epsilon \cdot \text{cost}_r)/(ha_i) \rfloor (\geq x_i)$  items from  $L_i$ . Since our selection is from large to small and the optimal solution with the rejected list  $U$  is *small-packed*,  $U_i \subseteq L_i \cap L'_r$ . And

$$p(L_i \cap L'_r) \leq \frac{(k_i + 1) \cdot \epsilon \cdot \text{cost}_r}{h} \leq p(U_i) + \frac{\epsilon \cdot \text{cost}_r}{h}.$$

By cases i) and ii), we have  $U \subseteq L'_r$  and  $p(L'_r) \leq p(U) + \epsilon \text{cost}_r$ . Since  $p(U) \leq \text{cost}_r \leq (1 + O(\epsilon))p(U) + 1$ , the lemma follows.  $\square$

We are now ready to prove our main result.

**Theorem 1.** *Our algorithm is an APTAS with time complexity  $O(n^{\epsilon^{-2}})$ .*

*Proof.* Let  $L$  be the input list and  $L_b \subseteq L$  be the list in which all the items have the rejection cost at least  $1/n$ . So,

$$OPT(L_b) \leq OPT(L) \leq OPT(L_b) + 1. \quad (1)$$

Let  $L_b = L_r \cup L_p$ , where  $L_r$  ( $L_p$ ) is the rejected (packed) list in some optimal solution of  $L_b$ . Then

$$OPT(L_b) = OPT_p + p(L_r), \quad (2)$$

where  $OPT_p$  is the number of bins used for  $L_p$  and  $p(L_r)$  is the total cost for the rejected list  $L_r$ . By Lemma 3,

$$p(L_r) \leq \text{cost}_r \leq (1 + \epsilon)p(L_r) + 1. \quad (3)$$

Lemma 4 says that there are no items with a rejection cost larger than 1 in  $L_r$ . After rounding down all the items in  $L_r$  at phase 1 of step 2.2, we obtain a new list  $\bar{L}_r$ . Then

$$p(\bar{L}_r) \leq p(L_r) \leq (1 + \epsilon)p(\bar{L}_r).$$

Combining the latter inequality with (3) we obtain,

$$p(\bar{L}_r) \leq \text{cost}_r \leq (1 + O(\epsilon))p(\bar{L}_r) + 1.$$

By Lemma 8,

$$p(\bar{L}'_r) \leq (1 + O(\epsilon))p(\bar{L}_r) + 1 \leq (1 + O(\epsilon))p(L_r) + 1, \quad (4)$$

where  $\bar{L}'_r$  is the rejected list guessed in step 2.2. Let  $L'_r$  be the rejected list before we round down list  $\bar{L}'_r$ . Then

$$p(L'_r) \leq (1 + \epsilon)p(\bar{L}'_r) \leq (1 + O(\epsilon))p(L_r) + 1 + \epsilon. \quad (5)$$

Let  $A$  be our algorithm and  $A(L)$  be the total cost by our algorithm. By (5), Lemma 2 and (2),

$$\begin{aligned} A(L) &\leq (1 + \epsilon)OPT_p + O(\epsilon^{-2}) + (1 + O(\epsilon))p(L_r) + 1 + \epsilon, \\ &\leq (1 + O(\epsilon))OPT(L) + O(\epsilon^{-2}). \end{aligned}$$

As for the time complexity, in step 1, by Lemma 3, our algorithm takes  $O(\epsilon^{-1}n \ln n)$  time, in step 2, by Lemma 6, our algorithm takes  $O(n^{O(\epsilon^{-2})})$  time, in step 3, our algorithm takes  $O(\epsilon^{-8}n \log n)$  time [9]. Then the time complexity of our algorithm is  $O(n^{O(\epsilon^{-2})})$ .  $\square$

### 3 An APTAS for variable-sized bin packing with rejection

In variable-sized bin packing, we are given a set of items  $L$  and a set of available bin sizes  $B$ . (As in [7, 11] we assume, without loss of generality, that the largest bin size in  $B$  is 1.) The object is to minimize the sum of the sizes of the bins used. It is clear that this problem is a generalization of the bin packing problem. If each item has both a size and a rejection cost associated with it and the object is to minimize the total cost, then the problem is the variable-sized bin packing with rejection.

In this section, we show that our approach can be extended to the variable-sized bin packing problem with rejection. This follows easily in the following way:

- The rounding technique in step 1 is still available, i.e, the guessed packing cost and rejection cost are near the optimal costs. (See Lemma 10.)
- The technique of rounding down the rejection cost in phase 2 of step 2 is still available, since the rounding is independent of the the packing.
- As with Lemma 2, there is an APTAS for variable-size bin packing as shown in Lemma 9.

**Lemma 9.** [11] *For the variable-sized bin packing problem, there is a fully polynomial time algorithm with the solution at most*

$$(1 + \epsilon)OPT + O(\epsilon^{-4}),$$

where  $OPT$  is the optimal value and  $\epsilon > 0$  is sufficiently small.

Since every item can be packed into one bin of size 1, the optimal value for  $n$  items is at most  $n$ . So  $OPT_p \leq n$  and  $OPT_r \leq n$ , where  $OPT_p$  ( $OPT_r$ ) is the total packed (rejection) cost in some optimal solution for variable-sized bin packing with rejection. Then the rounding technique in step 1 is still available and we can prove the following lemma:

**Lemma 10.** *By the same approach in step 1, there is a packed cost  $cost_p$  such that*

$$cost_p - 1 \leq OPT_p \leq cost_p.$$

In our algorithm for bin packing with rejection, if we replace the APTAS for bin packing in [9] with the one for variable bin packing in [11], then we get an algorithm for the variable-sized bin packing with rejection. We can see Lemmas 4, 7 and 8 still hold for the variable-sized case. Using the similar proof in Theorem 1, we can prove the following theorem.

**Theorem 2.** *There is an APTAS for variable-sized bin packing with rejection in time  $O(n^{\epsilon^{-2}})$ .*

## 4 Concluding remarks

In this paper, we give a fast APTAS for bin packing with rejection and show the approach can be extended to variable-sized bin packing with rejection. The existence of an *asymptotic fully polynomial time approximation scheme* for bin packing with rejection is open.

## References

1. W. Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. Sgall, L. Stougie. Multiprocessor scheduling with rejection. *SIAM Journal on Discrete Mathematics* 13(1):64–78, 2000.
2. C. Chekuri and S. Khanna. A polynomial time approximation scheme for the multiple knapsack problem, *SIAM J. Comput.* 35(3): 713-728, 2005.
3. E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: a survey. In *D. Hochbaum, Editor, Approximation algorithms for NP-hard problems*, 46–93. PWS Publishing, Boston, 1996.
4. G. Dósa and Y. He. Bin packing problems with rejection penalties and their dual problems. *Information and Computation* 204(5):795–815, 2006.
5. L. Epstein. Bin packing with rejection revisited. In *Proc. 4th Workshop on Approximation and Online Algorithms (WAOA)*, 146-159, 2006.
6. W. Fernandez de la Vega and G. Lueker. Bin packing can be solved within  $1 + \epsilon$  in linear time. *Combinatorica* 1:349–355, 1981.
7. D.K. Friesen and M.A. Langston. Variable sized bin packing. *SIAM Journal on Computing*, 15(1):222-230, 1986.
8. H. Hoogeveen, M. Skutella and G. Woeginger. Preemptive scheduling with rejection. *Mathematical Programming, Ser. B*, 94(2-3):361–374, 2003.
9. N. Karmarkar and R.M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem, In *Proc. 23rd Annual IEEE Symp. Found. Comput. Sci.*, 312-320, 1982.
10. H. Kellerer. A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem, In *Proc. RANDOM-APPROX 99*, 51-62, 1999.
11. F.D. Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM Journal on Computing*, 16(1):149-161,1987.