

A Fast Asymptotic Approximation Scheme for Bin Packing with Rejection

Wolfgang Bein¹, José R. Correa², Xin Han³

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

bein@cs.unlv.edu

² School of Business, Univesidad Adolfo Ibáñez, Santiago, Chile

correa@uai.cl

³ School of Informatics, Kyoto University, Kyoto 606-8501, Japan

hanxin@kuis.kyoto-u.ac.jp

Abstract

“Bin packing with rejection” is the following problem: 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. We show that bin packing with rejection can be reduced to n multiple knapsack problems and, based on techniques for the multiple knapsack problem, we give a fast asymptotic polynomial time approximation scheme, “Reject&Pack”, with time complexity $O(n^{O(\varepsilon^{-2})})$. This improves a recent approximation scheme given by Epstein, which has time complexity $O(n^{O((\varepsilon^{-4})^{\varepsilon^{-1}})})$. We also show that Reject&Pack can be extended to variable-sized bin packing with rejection and give an asymptotic polynomial time approximation scheme.

Keywords: Approximation Scheme; Bin Packing; Knapsack Problem; Rejection Penalty.

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 for over four decades (see [3] for a detailed survey).

Interestingly, although bin packing is \mathcal{NP} -hard, it was one of the first problems for which approximation schemes were designed. We say approximation algorithm \mathcal{A} for \mathcal{NP} -hard optimization problem Π is a *polynomial time approximation scheme* (PTAS), if for any fixed $\varepsilon > 0$, its run time is polynomial in the size of the instance of the problem, and its cost is within a factor $1 + \varepsilon$ of the optimal cost. Furthermore, we say an approximation algorithm \mathcal{A} for Π is an *asymptotic polynomial time approximation scheme* (APTAS), if the previous cost bound holds asymptotically. That is, if for any fixed $\varepsilon > 0$, the run time of \mathcal{A} is polynomial

and its cost is at most $(1+\varepsilon)\cdot OPT+O(1)$, where OPT denotes the optimal cost. Asymptotic polynomial time approximation schemes for bin packing were obtained early on by Fernandez de la Vega and Luecker [6] and by Karmarkar and Karp [12]. However, bin packing does not admit a PTAS and there cannot be any approximation algorithm with approximation ratio better than $3/2$ unless $\mathcal{P} = \mathcal{NP}$ (see Chapter 9 of [18]).

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, 9]). 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 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 list¹ 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 \mathcal{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 offline results. In particular, they give an algorithm with asymptotic approximation ratio of $3/2$. Epstein [5] gave the first asymptotic polynomial time approximation scheme for the bin packing with rejection problem based on classical bin packing techniques.

A problem somewhat similar to bin packing (and probably as important) is the *knapsack problem*: Given a knapsack with capacity $c > 0$ and a set of items with associated profits and weights, find a subset of items such that the total size is bounded by c and the total profit is maximized. A pseudo-polynomial time algorithm and a PTAS for the knapsack problem were first given by Ibarra and Kim [10]. Lawler [15] improved the running time of their algorithm to $O(n \log 1/\varepsilon + 1/\varepsilon^4)$. A natural generalization of the knapsack problem is a variant with multiple knapsacks, the so-called *multiple knapsack problem*. Kellerer [13] was the first to give a PTAS for the special case of this problem where all the knapsacks have identical capacity [13]. Later, Chekuri and Khanna [2] obtained a PTAS for the general multiple knapsack problem. We refer the reader to the book by Kellerer, Pferschy and Pisinger [14] for a comprehensive treatment of knapsack problems.

Our contribution. Borrowing techniques for the multiple knapsack problem [2, 13], we construct an APTAS for bin packing with rejection which is more efficient than that of Epstein. More 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 prob-

¹We use the term “list” and “set” synonymously.

lem, we give a fast asymptotic polynomial time approximation scheme with time complexity $O(n^{O(\varepsilon^{-2})})$ thereby improving the current best $O(n^{O((\varepsilon^{-4})^{\varepsilon^{-1}})})$ scheme of [5]. Furthermore, we extend our scheme to variable-sized bin packing with rejection and give an APTAS for this problem as well.

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. We give the main idea for our APTAS:

- First, guess packing cost $cost_p$ and rejection cost $cost_r$ such that the two values are not much larger than the corresponding values in an optimal solution.
- Then, based on those two guessed values, guess a sublist L_r of the input list L such that the total rejection cost in L_r is bounded by $(1 + O(\varepsilon)cost_r)$ and it is feasible to pack all items of $L - L_r$ into $cost_p$ many bins.
- Finally, call a bin packing APTAS to pack all items of $L - L_r$.

The key point is that we can guarantee that all guessing steps can be done in time polynomial in n .

One of the main techniques of our paper is to combine geometric rounding with careful enumeration. Another idea employed in our paper is to further combine these with techniques for multiple knapsack. We suspect that the results and ideas presented in this paper could prove useful to tackle other packing problems with item selections, *i.e.*, problems in which items are not fixed a priori. Further applications could be in scheduling, see *e.g.* [17].

2 An APTAS for the bin packing with rejection

We first show that in order to solve an instance of bin packing with rejection where the input list is of size n , we only need to solve at most n multiple knapsack problems.

Given an input list L , the value $OPT(L)$ of any optimal solution can be written as follows:

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

where $OPT_p(L)$ is the cost for the packing (*i.e.*, the number of bins used), and $OPT_r(L)$ is the rejection cost of the solution. We use L_p to denote the packed items in the solution, and L_r to denote the rejected items in the solution; we call L_p the *packed list* and L_r the *rejected list*.

Definition 1 For any list S we denote the sum of rejection costs in S by $r(S)$. In other words, $r(S) = \sum_{i \in S} r_i$.

Lemma 1 If OPT_p is fixed then bin packing with rejection is equivalent to the multiple knapsack problem of packing the input list into OPT_p unit bins in order to maximize the total profit in the bins, where the profit of an item equals its rejection penalty in the packing case.

Proof. Let $OPT_p = i$. Hence,

$$\begin{aligned} OPT(L) &= OPT_p(L) + OPT_r(L) \\ &= i + r(L_r) \\ &= i + r(L) - r(L_p). \end{aligned}$$

Thus, $OPT(L)$ is minimized when $r(L_p)$ is maximized. Note that maximizing $r(L_p)$ is equivalent to selecting a sublist L_p from the input list L to be packed into i bins to maximize the total rejection cost of the bins. Hence, the knapsack problem (of packing L into i knapsacks) is solved, exactly when the the original problem is solved. \square

Lemma 1 suggests the following algorithm:

Algorithm 1 An intuitive algorithm, which is not an APTAS.

- Guess OPT_p from 1 to n , i.e., the number of bins to be used.
 - Consider rejection costs to be profits in a multiple knapsack problem. Then call the PTAS of [2, 13] for the multiple knapsack problem and pack the items into OPT_p many bins (knapsacks). Unpacked items make up the rejected list R .
 - Output $\min\{OPT_p + r(R)\}$ over all guessed values OPT_p , $OPT_p = 1 \dots n$.
-

Unfortunately, Algorithm 1 does not give an approximation scheme, not even an APTAS. This is easily seen as follows: In Algorithm 1, the packing cost is OPT_p , i.e., the number of bins used. Let X be the maximum profit (rejection cost) which is packed into bins with OPT_p . The total profit packed by the PTAS of [2, 13] is $(1 - \varepsilon)X$. Then the total cost for our algorithm is

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

The value of εX can be substantial, even larger than $OPT(L)$, so one cannot guarantee that Algorithm 1 produces a solution near the optimal cost.

However, an APTAS can be constructed in another way. The intuition is to focus on the rejected list instead of the packed list: We guess a rejected list such that the total cost in it is not significantly larger than that of some optimal solution and all the remaining items can be packed in the bins we guessed.

Before going into further details we recall the following classical bin packing result of Karmarkar and Karp [12]:

Theorem 1 *For an instance of the bin packing problem with n items, given $\varepsilon > 0$, there is an algorithm which runs in time polynomial in n and $1/\varepsilon$ and finds a solution for the instance which uses at most*

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

bins, where OPT is the optimal number of bins for the instance.

Our scheme, which we call REJECT&PACK, has three major steps. First, guess both OPT_p and OPT_r . (Recall, OPT_p and OPT_r denote the packing cost and the rejection cost of some optimal solution.) Then, based on these two values, guess a packed list and a rejected list. Finally, use the Karmarkar-Karp algorithm of Theorem 1 to deal with the packed list and reject the other items.

Algorithm 2 REJECT&PACK.

Step A: Guess the packing cost $cost_p$ and the rejection cost $cost_r$ such that

$$cost_p = OPT_p \text{ and } OPT_r \leq cost_r \leq (1 + \varepsilon)OPT_r + 1.$$

Step B: Guess a rejected list L'_r such that

$$cost_r \leq r(L'_r) \leq (1 + O(\varepsilon))cost_r$$

and the remaining items $L - L'_r$ can be packed into $cost_p$ many bins.
Reject all items in L'_r .

Step C: Call APTAS [12] to pack $L - L'_r$. Output the bins used and $r(L'_r)$.

2.1 Guessing the rejection cost and the packing cost

A trivial feasible solution for any instance of the bin packing with rejection problem is to pack all items without rejecting any. Thus, the cost of an optimal solution is at most n , where n is the size of the input list. Therefore OPT_p and OPT_r are bounded by n as well. This implies that there are at most $n + 1$ possible values for OPT_p . On the other hand, if $OPT_r \leq 1$, a good estimate is given by $cost_r = 1$, at least in the asymptotic sense. Otherwise, if $(1 + \varepsilon)^i \leq OPT_r < (1 + \varepsilon)^{i+1}$ for some $i \geq 0$, then $(1 + \varepsilon)^{i+1}$ is a good estimate for OPT_r , so we set $cost_r = (1 + \varepsilon)^{i+1}$, where $\varepsilon > 0$ is the error bound. Since $OPT_r \leq n$, we can obtain

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

where the last inequality follows from the fact that $\ln(1 + \varepsilon) \geq \varepsilon - \varepsilon^2/2$ and $\varepsilon \leq 1/2$. Thus the following lemma holds:

Lemma 2 *Let OPT_p be the packing cost and OPT_r be the rejection cost of some optimal solution for an instance of the bin packing with rejection problem. Then the estimates $cost_r$ and $cost_p$ can be guessed in time $O(\varepsilon^{-1}n \ln n)$ and they satisfy:*

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

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

2.2 Guessing the rejected list and the packed list

We now describe how to guess rejected and packed items with total cost near the optimal values using the guesses $cost_r$ and $cost_p$ from the previous step. As before, we denote the guessed rejected list by L'_r and the guessed packed list by L'_p . Furthermore we note:

Lemma 3 *Given input list L , if any item $i \in L$ has rejection cost $r_i > 1$ then there is an optimal solution where item i is 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 are now ready to detail Steps B and C of REJECT&PACK:

Guessing the rejected list and the packed list.
<p>I. Place every item with rejection cost larger than 1 into list L'_p. Place all items with rejection cost smaller than $1/n$ into the rejected list L'_r.</p> <p>II. The remaining items have rejection costs in the range $[1/n, 1]$. There are three phases to assign all the remaining items into either L'_p or L'_r:</p> <ul style="list-style-type: none"> i) Round down each item's rejection cost to obtain $O(\varepsilon^{-1} \ln n)$ sublists such that in each sublist all the items have the same rejection cost. ii) Guess the rejected items in each sublist and place them into L'_r such that L'_r has its rejection cost in $[cost_r, (1 + O(\varepsilon)cost_r)]$. iii) Place all remaining items into L'_p. Test whether packing L'_p into $cost_p$ many bins is feasible.

Below are the details of the three phases in Step II; they are referred to as “rounding down”, “guessing the rejected list” and “testing the packed list.”

Rounding down.
<p>Given an item with a rejection cost r, round r down to \bar{r} for some integer $i \geq 0$ such that</p> $\bar{r} = \frac{(1 + \varepsilon)^i}{n} \leq r < \frac{(1 + \varepsilon)^{i+1}}{n},$ <p>where ε is the given error bound.</p> <p>Since $r \leq 1$, $\frac{(1 + \varepsilon)^i}{n} \leq 1$ and $i \leq \frac{\ln n}{\ln(1 + \varepsilon)} \leq O(\varepsilon^{-1} \ln n)$. Obtain $h = O(\varepsilon^{-1} \ln n)$ many sublists L_i. Each item in L_i has the same rejection cost $\frac{(1 + \varepsilon)^i}{n}$.</p>

Let U be the rejected items in some optimal solution of $\cup_i L_i$. Let $U_i = L_i \cap U$ and let $r(U_i)$ be the total rejection cost in U_i . Items from L_i are selected as follows:

Guessing the rejected list.

Guess approximately the values $r(U_i)$ for $0 \leq i \leq h$. For each i , guess an integer $k_i \in [0..h/\varepsilon]$ such that

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

Guess the rejected items in each sublist L_i . Let a_i be the rejection cost of one item in L_i , i.e., $a_i = \frac{(1+\varepsilon)^i}{n}$. For each k_i for $0 \leq i \leq h$, order all items in L_i in the order of non-decreasing size values.

If $a_i > \varepsilon \cdot \text{cost}_r/h$ then pick the largest $\lceil k_i(\varepsilon \cdot \text{cost}_r/h)/a_i \rceil$ items from L_i and put them into L'_r .

Else pick the largest $\lfloor (k_i + 1)(\varepsilon \cdot \text{cost}_r/h)/a_i \rfloor$ items from L_i and put them into L'_r .

In Subsection 2.3 we show that the number of candidates for k_i can be bounded by $O(n^{\varepsilon^{-2}})$, thus there are a polynomial number of rejected lists.

Testing the packed list.

Each time, after selecting the rejected item from L_i , put the remaining items in L_i into the packed list L'_p . Then use APTAS [12] to pack the items in L'_p into $(1 + \varepsilon)\text{cost}_p + O(\varepsilon^{-2})$ bins. If all items in L'_p can be packed then return L'_p and L'_r . Else try another tuple (k_1, \dots, k_h) .

2.3 Analysis

We now prove that the number of all candidate h -tuples (k_1, \dots, k_h) can be bounded by $O(n^{O(1/\varepsilon^2)})$, which is polynomial in n . To this end, we first invoke a lemma due to Chekuri and Khanna [2]. (We include the short proof for completeness.)

Lemma 4 *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 uses 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 $(g-1)!$ can be approximated by $\sqrt{2\pi(g-1)}((g-1)/e)^{g-1}$. Hence, $f = O((\alpha e)^{g-1}) = O(e^{\alpha g})$. \square

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

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

where the last inequality follows from the fact that $r(U) \leq OPT_r \leq cost_r$. By the above observation, $\sum_i k_i \leq h/\varepsilon = O(\ln n/\varepsilon^2)$, we obtain the following lemma.

Lemma 5 *Let h be an integer with $h \leq 2\varepsilon^{-1} \ln n$. Then the number of h -tuples (k_1, \dots, k_h) with $\sum_i k_i \leq h/\varepsilon$ is $O(n^{O(\varepsilon^{-2})})$.*

Proof. We apply the bound from Lemma 4. Thus, we have that $\alpha = (1 + 1/\varepsilon)$ and $g = 2\varepsilon^{-1} \ln n$; hence we get an upper bound $e^{2\varepsilon^{-1}(1+1/\varepsilon)\ln n}$. \square

For our analysis we introduce the notion of a “*small-packed*” solution:

Definition 2 (Small-packed) *Given a solution of packed list L_p and rejected list L_r , if for any two items $i \in L_p$ and $j \in L_r$ with $r_i = r_j$ it follows that $s_i \leq s_j$, then we say the solution is small-packed, otherwise not small-packed.*

Thus, if there exist packed item i and rejected item j such that $r_i = r_j$, and $s_i > s_j$ then such a solution is not small packed. In that case one can exchange items i and j , i.e., pack j and reject i , to get another solution without increasing the total cost. Repeatedly applying such exchanges, gives the the following lemma:

Lemma 6 *There exists a small-packed optimal solution for the bin packing with rejection problem.*

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

Lemma 7 *Assume that every item has rejection costs equal to $\frac{(1+\varepsilon)^i}{n} \leq 1$ for some integer i , with $0 \leq i \leq h = O(\varepsilon^{-1} \ln n)$. Let U be the set of the rejected items in some optimal solution with $r(U) \leq cost_r \leq (1 + O(\varepsilon))r(U) + 1$. Then there exists a tuple (k_1, \dots, k_h) associated with the rejected list L'_r such that $U \subseteq L'_r$ and $r(L'_r) \leq (1 + O(\varepsilon))r(U) + 1$.*

Proof. Due to Lemma 6 we can assume without loss of generality that U is a set of the rejected items in some optimal solution with the small-packed property. Then for all i , we define

$$k_i = \lfloor r(U_i)h/(\varepsilon \cdot cost_r) \rfloor.$$

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 $a_i \cdot x_i = r(U_i)$.

- (i) $a_i > \varepsilon \cdot cost_r/h$. We prove our selection from L_i is the same as $U_i = U \cap L_i$, i.e., $U_i = L_i \cap L'_r$. By the definition $k_i = \lfloor r(U_i)h/(\varepsilon \cdot cost_r) \rfloor$, so

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

Then

$$r(U_i) - \frac{\varepsilon \cdot cost_r}{h} < \frac{k_i \cdot \varepsilon \cdot cost_r}{h} \leq r(U_i)$$

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

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

So we select $\lceil (k_i \cdot \varepsilon \cdot cost_r)/(ha_i) \rceil (= x_i)$ 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) $a_i \leq \varepsilon \cdot \text{cost}_r/h$. We prove that $U_i \subseteq (L_i \cap L'_r)$ and $r(L_i \cap L'_r) \leq r(U_i) + \frac{\varepsilon \cdot \text{cost}_r}{h}$. Again, by definition $k_i = \lfloor r(U_i)h/(\varepsilon \cdot \text{cost}_r) \rfloor$, so

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

Then

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

So we select $\lfloor ((k_i + 1)\varepsilon \cdot \text{cost}_r)/(ha_i) \rfloor (\geq x_i)$ 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

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

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

We are now ready to prove our main result.

Theorem 2 REJECT&PACK is an APTAS for the bin packing with rejection problem with time complexity $O(n^{\varepsilon^{-2}})$.

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

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

where OPT_p is the number of bins used for L_p and $r(L_r)$ is the total cost for the rejected list L_r . By Lemma 2,

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

Lemma 3 says that there are no items with a rejection cost larger than 1 in L_r . After rounding down (Step III) all items in L_r we obtain a new list \bar{L}_r . Then

$$r(\bar{L}_r) \leq r(L_r) \leq (1 + \varepsilon)r(\bar{L}_r).$$

Combining the latter inequality with (3) we obtain,

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

By Lemma 7,

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

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

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

Let $A(L)$ be the total cost obtained by of REJECT&PACK. By (5), Theorem 1 and (2),

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

As for the time complexity, in Step A, by Lemma 2, REJECT&PACK takes $O(\varepsilon^{-1}n \ln n)$ time. In Step B, by Lemma 5, it takes $O(n^{O(\varepsilon^{-2})})$ time. In Step C, it takes $O(\varepsilon^{-8}n \log n)$ time [12]. Therefore, the time complexity of REJECT&PACK is $O(n^{O(\varepsilon^{-2})})$. \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, 16] we assume, without loss of generality, that the largest bin size in B is 1.) The objective 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 objective is to minimize the total cost, then the problem is the variable-sized bin packing with rejection. We recall here the APTAS of Murgolo [16]:

Theorem 3 *For an instance of the variable-sized bin packing problem with n items, given $\varepsilon > 0$, there is an algorithm which runs in time polynomial in n and $1/\varepsilon$ and finds a solution for the instance with value*

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

where OPT is the value of an optimal solution for the instance.

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 A is still available, i.e, the guessed packing cost and rejection cost are near the optimal costs: Since every item can be packed into one bin of size 1, the optimal value for n items is at most n . Thus $OPT_p \leq n$ and $OPT_r \leq n$, where OPT_p (OPT_r) is the total packing (rejection) cost in some optimal solution for variable-sized bin packing with rejection.
- The technique of rounding down the rejection cost in Step IIi) is still available, since the rounding is independent of the the packing.
- As with Theorem 1, there is an APTAS for variable-size bin packing as shown in Theorem 3.

Thus if in REJECT&PACK we replace the APTAS for bin packing in [12] with the one for variable bin packing in [16], then we get an algorithm for the variable-sized bin packing with rejection. Using adapted versions of Lemmas 3, 6, 7 and invoking Theorem 2, we can prove the following theorem.

Theorem 4 *There is an APTAS for variable-sized bin packing with rejection with time complexity $O(n^{\varepsilon^{-2}})$.*

4 Concluding remarks

As mentioned in the introduction, geometric rounding and careful enumeration combined with techniques for bin packing and multiple knapsack could prove useful to tackle other hard combinatorial problems.

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

Acknowledgments

The work of the first author was done while visiting Japan as Kyoto University Visiting Professor during the 2006 Winter semester and on sabbatical from the University of Nevada, Las Vegas. The research of the second author was supported in part by CONICYT through grants FONDECYT 10600035 and ACT08. The third author acknowledges support from NSFC (10231060).

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. Preliminary version in proceedings of COCOON 2005, Springer LNCS 3595, 885–894, 2006.
- [5] L. Epstein. Bin packing with rejection revisited. In *proceedings of the 4th Workshop on Approximation and Online Algorithms (WAOA)*, 2006, 146–159.

- [6] W. Fernandez de la Vega and G. Lueker. Bin packing can be solved within $1 + \varepsilon$ 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] M.R. Garey and D.S. Johnson. Computers and intractability: a guide to the theory of NP-completeness. *W.H. Freeman and Co., San Francisco*, 1979.
- [9] H. Hoogeveen, M. Skutella and G. Woeginger. Preemptive scheduling with rejection. *Mathematical Programming, Ser. B* 94(2-3):361–374, 2003.
- [10] O.H. Ibarra, C.E. Kim. Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems. *J. ACM* 22(4):463-468, 1975.
- [11] D.S. Johnson and M. R. Garey. A $71/60$ theorem for bin packing. *J. Complexity*, 1(1): 65-106, 1985.
- [12] 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.
- [13] H. Kellerer. A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem. *RANDOM-APPROX* 51-62, 1999.
- [14] H. Kellerer, U. Pferschy and D. Pisinger. *Knapsack problems*. Springer, Berlin, 2004.
- [15] E.L. Lawler. Fast Approximation Algorithms for Knapsack Problems. *Math. Oper. Res.* 4(4):339-356, 1979.
- [16] F.D. Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM Journal on Computing*, 16(1):149-161,1987.
- [17] H. Shachnai and T. Tamir. Polynomial Time Approximation Schemes - A Survey. In *T. F. Gonzalez, editor, Handbook of Approximation Algorithms and Metaheuristics*, Chapman & Hall , Boca Raton, 2007.
- [18] V. Vazirani. *Approximation Algorithms*. Springer, Berlin, 2001.