

# 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> School of Computer Science, University of Nevada, Las Vegas, NV 89154, USA

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

<sup>3</sup> School of Informatics, Kyoto University, Kyoto 606-8501, Japan

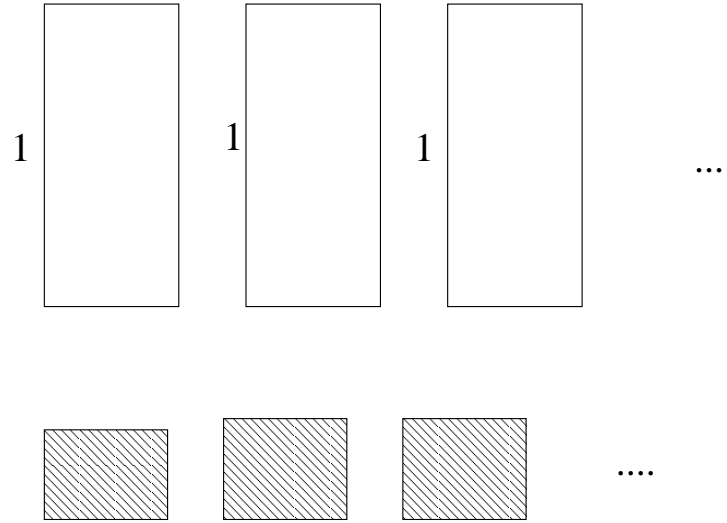
# Outline

1. Definitions and Contributions
2. Bin Packing with Rejection vs Knapsack Problem
3. Key ideas in our algorithm for BPR (Bin Packing with rejection) problem
4. Open questions

# Outline

1. Definitions and Contributions
2. Bin Packing with Rejection vs Knapsack Problem
3. Key ideas in our algorithm for BPR (Bin Packing with rejection) problem
4. An open question

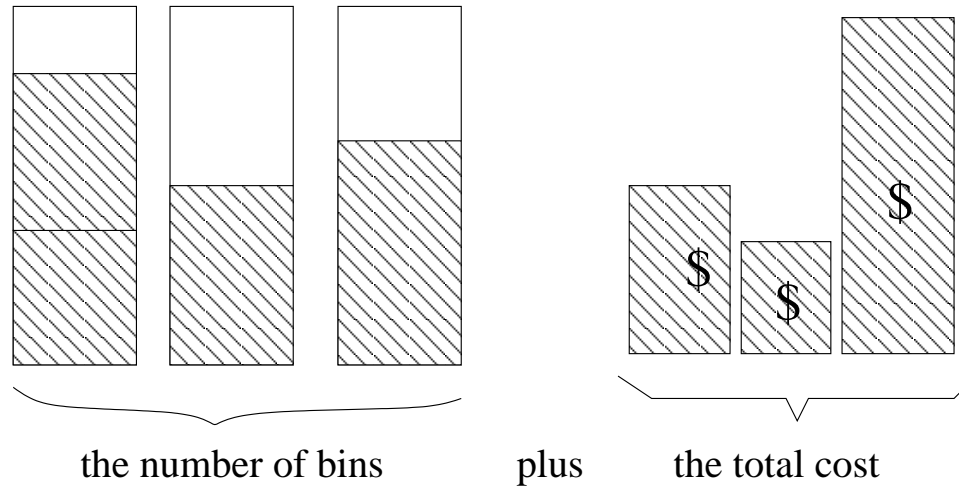
# One dimensional bin packing problem



- **Input:** a collection of one dimensional items with size at most 1
- **Output:** minimize the number of bins required.

Note that: This problem is NP-hard.

# Bin packing with rejection



- **Input:** a collection of items with a size and a rejected cost
- **Output:** minimize the number of bins used plus the total cost of all the rejected items.

# Studies on NP-hard problems

It is likely that there do not exist polynomial-time algorithms for NP-hard problems.

- Approximation algorithm study.
- Exact algorithm study.

# Approximation ratios

Let  $P$  be an optimization (minimization) problem. The *approximation ratio* of algorithm  $A$  is defined as:

$$R_A = \max_I \left\{ \frac{A(I)}{Opt(I)} \right\} \text{ i.e., } A(I) \leq R_A \times Opt(I).$$

The asymptotic approximation ratio  $R_A^\infty$  of algorithm  $A$  is defined as:

$$A(I) \leq R_A^\infty \times Opt(I) + C,$$

where  $I$  ranges over the set of all problems instances and  $C$  is a constant.

# PTAS and APTAS

For a given constant  $\epsilon$ ,

- if  $R_A = (1 + \epsilon)$  then  $A$  is PTAS, i.e.,

$$A(I) \leq (1 + \epsilon) \cdot \text{Opt}(I).$$

- if  $R_A^\infty = (1 + \epsilon)$  then  $A$  is APTAS, i.e.,

$$A(I) \leq (1 + \epsilon) \cdot \text{Opt}(I) + C.$$

PTAS = *polynomial time approximation scheme*.

APTAS = *asymptotic PTAS*.

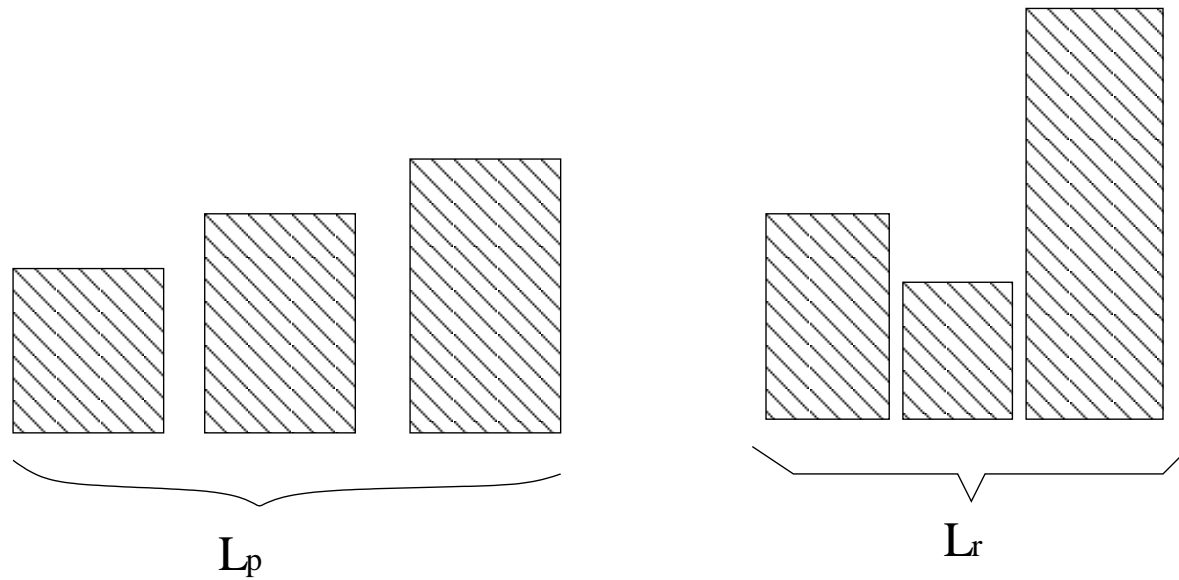
# Previous research on BPR

- First studied by He and Dósa [2005].
- APTAS was given by Epstein [2006].

# Contributions on BPR

- BPR is equivalent to KP (Knapsack problem).
- We provided a **different** APTAS which turns out to be more efficient than that of Epstein.
- Furthermore we extended the scheme to variable-sized bin packing with rejection.

# Notations



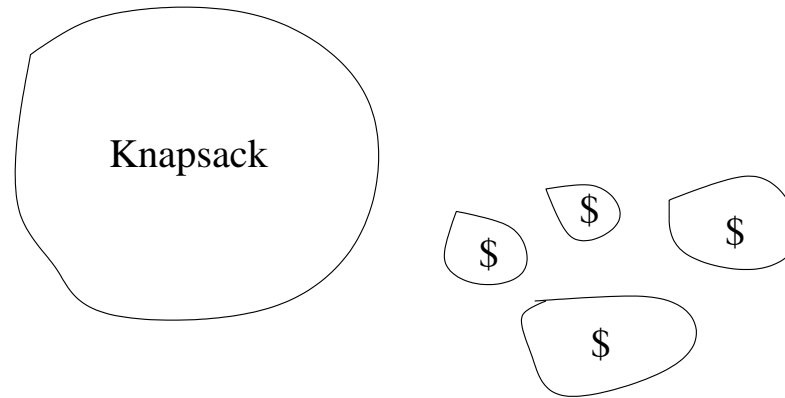
$$L = L_p \cup L_r,$$

$$OPT(L) = Cost(L_p) + Cost(L_r) = OPT_p(L) + OPT_r(L).$$

# Outline

1. Definitions and Contributions
2. Bin Packing with Rejection vs Knapsack Problem
3. Key ideas in our algorithm for BPR (Bin Packing with rejection) problem
4. Open questions

# Knapsack problem



- **Input:** a knapsack with a capacity  $B > 0$  and a set of items associated with profits and weights,
- **Output:** find a subset of the items whose total size is bounded by  $B$  and total profit is maximized.

Note that: this problem is NP-hard too. If the number of knapsack is  $m$  then it is MKP (Multiple Knapsack Problem).

# BPR $\Leftrightarrow$ MKP

$L = L_p \cup L_r$ ,  $p(L)$  is the total rejection cost in list  $L$ .

Assume  $cost(L_p) = i$ ,

$$\begin{aligned} OPT(L) &= cost(L_p) + cost(L_r) \\ &= i + p(L_r) \\ &= i + p(L) - p(L_p) \end{aligned}$$

We know  $OPT_p$  can be guessed exactly in  $O(n)$  time from  $[0.. n]$ . So, the BPR problem is equivalent to the multiple knapsack problem.

# Outline

1. Definitions and Contributions
2. Bin Packing with Rejection vs Knapsack Problem
3. Key ideas in our algorithm for BPR (Bin Packing with rejection) problem
4. Open questions

# The basic ideas for our APTAS

- Divide  $L$  into two parts  $L'_p$  and  $L'_r$ .
- $OPT_r(L) \leq p(L'_r) \leq (1 + \epsilon)OPT_r(L) + 1$ .
- $L'_p \Rightarrow OPT_p(L)$  bins.

# Main steps

- Guess the rejection cost and packing cost respectively.
- Guess the rejected list.
- Pack the packing list by bin packing algorithms.

Notations:  $cost_p$ ,  $cost_r$  are for the guessing packing and rejection costs, respectively.

# Guess packing cost $cost_p$

- Guess  $cost_p$  from set  $\{0, 1, 2, \dots, n\}$ .

So,  $cost_p$  can be guessed in time  $O(n)$  such that

$$cost_p = cost(L_p).$$

# How to guess rejection cost $cost_r$ ?

Keep two issues in mind

- $0 \leq cost(L_r) \leq n$
- Our target is  $cost(L_r) \leq cost_r \leq (1 + \epsilon)cost(L_r) + 1$ , not  $cost_r = cost(L_r)$ .

# Guess $cost_r$

- Guess  $cost_r$  from set  $K = \{(1 + \epsilon), (1 + \epsilon)^2, \dots\}$

Since  $cost(L_r) \leq n$ , the size of set  $K$  is bounded by  $O\left(\frac{\ln n}{\ln(1+\epsilon)}\right)$ .

# Guess $cost_p$ and $cost_r$

So,  $cost_p$  and  $cost_r$  can be guessed in  $O(n \ln n)$  such that

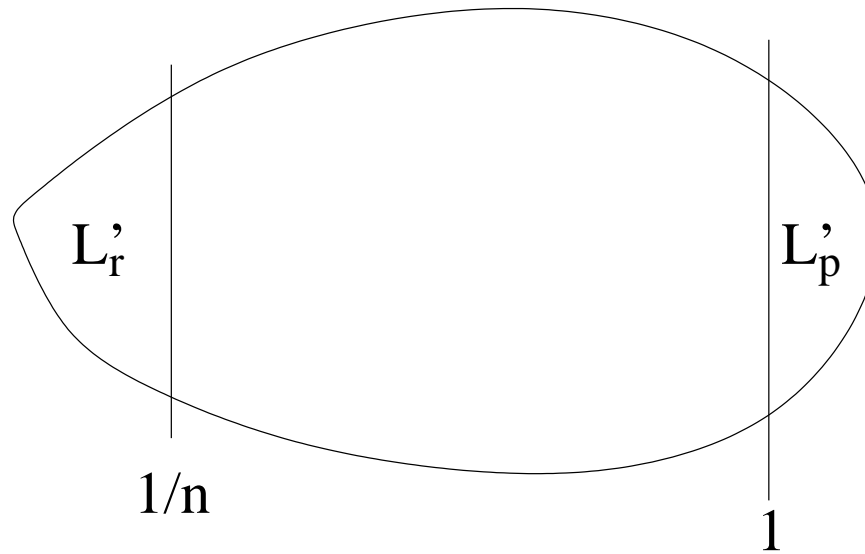
$$cost_p = OPT_p, \quad OPT_r \leq cost_r \leq (1 + \epsilon)OPT_r + 1.$$

# Main steps

- Guess the rejected cost and packing cost respectively.
- According to the costs, guess the rejected list.
- Pack the packing list by bin packing algorithms.

# Guessing the rejected list (1)

- Place every item with  $r > 1 \Rightarrow$  list  $L'_p$ .
- All items with  $r < 1/n \Rightarrow$  the rejected list  $L'_r$ .



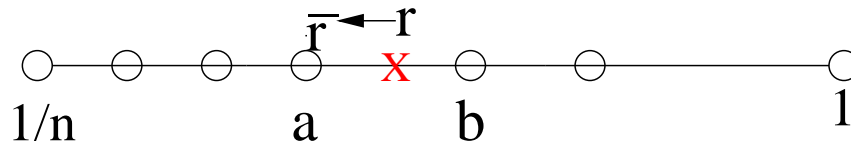
# Guessing the rejected list (2)

Consider the remaining items with  $r$  in  $[1/n, 1]$ .

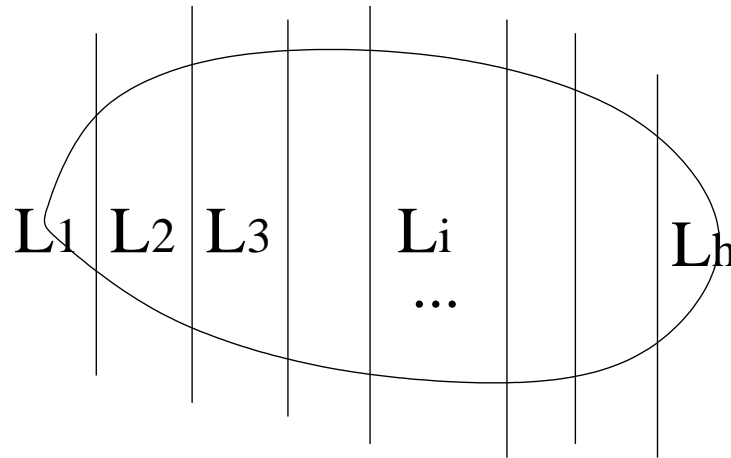
- Rounding down the rejection costs.
- Guessing rejected items.
- Testing the packed list.

# Rounding down

$$\text{If } \frac{(1 + \epsilon)^i}{n} \leq r < \frac{(1 + \epsilon)^{i+1}}{n} \text{ then } r \Leftarrow \frac{(1 + \epsilon)^i}{n}.$$



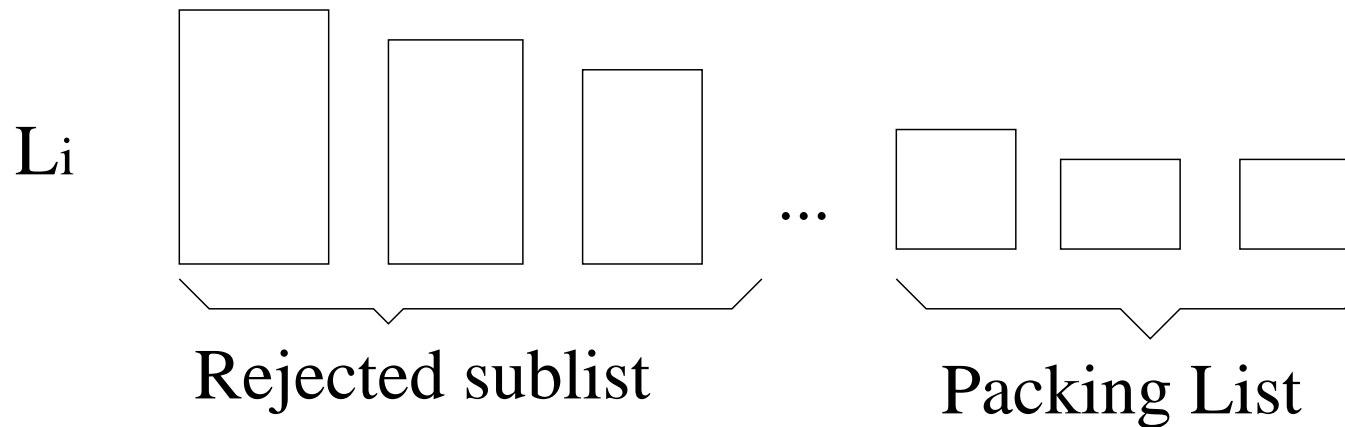
Then get  $h = O(\epsilon^{-1} \ln n)$  kinds of rejection costs and  $L_r$  is divided into  $h$  kinds of sublists.



And all items in  $L_i$  have  $\bar{r} = \frac{(1+\epsilon)^i}{n}$ .

# Guessing the rejected items from $L_i$

- Guess  $L_i$ 's contribution  $p(U_i)$ .
- Divide  $L_i$  into two parts.

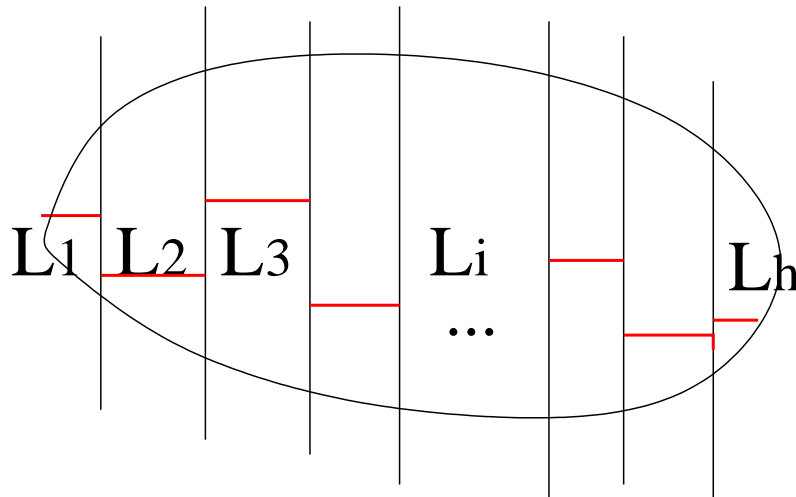


# Guessing the rejected items from $L_i$

- Guess  $L_i$ 's contribution

$$p(U_i) = \{k_i \times \frac{\epsilon \text{cost}_r}{h} \mid k_i = 0, \dots, h/\epsilon\} \text{ on } \text{cost}_r.$$

- Pick the largest  $p(U_i)/a_i$  items from  $L_i$  and put them into  $L'_r$ , where  $a_i = \frac{(1+\epsilon)^i}{n}$  is one item's cost in  $L_i$ .



# Technical results (Skipped)

- The number of all the candidates for  $p(U_i)$  can be bounded by  $O(n^{\epsilon^{-2}})$ .
- For the rejected list  $L'_r$ , we have

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

# Testing the packed list

$L'_p \Leftarrow L - L'_r$ . Then try to use APTAS to pack the items in  $L'_p$  into  $(1 + \epsilon)cost_p + O(\epsilon^{-2})$  bins.

- If “Yes” then return  $L'_p$  and  $L'_r$ .
- Else try another tuple  $(p(U_1), \dots, p(U_h))$ .

# Outline of our algorithm

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

# Theorems

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

# Outline

1. Definitions and Contributions
2. Bin Packing with Rejection vs Knapsack Problem
3. Key ideas in our algorithm for BPR (Bin Packing with rejection) problem
4. Open questions

# An open question

- Does BPR admit an AFPTAS or not?