

# A Sublinear-Time Approximation Scheme for Bin Packing

Tuğkan Batu<sup>1</sup>   Petra Berenbrink<sup>2</sup>   Christian Sohler<sup>3</sup>

<sup>1</sup>Department of Mathematics, London School of Economics

<sup>2</sup>School of Computing Science, Simon Fraser University

<sup>3</sup>Department of Computer Science, University of Paderborn

Bristol Algorithms Days, 4 March 2008

# Bin Packing Definition

## Definition (Bin Packing)

Given  $w_1, w_2, \dots, w_n$  such that  $0 < w_j \leq 1$ , find a packing of these items into minimum number of unit-size bins.

NP-hard.

$\alpha$ -approximation:  $OPT \leq \text{Alg.'s output} \leq \alpha \cdot OPT$

# Bin Packing Definition

## Definition (Bin Packing)

Given  $w_1, w_2, \dots, w_n$  such that  $0 < w_j \leq 1$ , find a packing of these items into minimum number of unit-size bins.

NP-hard.

$\alpha$ -approximation:  $OPT \leq \text{Alg.'s output} \leq \alpha \cdot OPT$

# Bin Packing Definition

## Definition (Bin Packing)

Given  $w_1, w_2, \dots, w_n$  such that  $0 < w_j \leq 1$ , find a packing of these items into minimum number of unit-size bins.

NP-hard.

$\alpha$ -approximation:  $\text{OPT} \leq \text{Alg.'s output} \leq \alpha \cdot \text{OPT}$

# Bin Packing Definition

## Definition (Bin Packing)

Given  $w_1, w_2, \dots, w_n$  such that  $0 < w_j \leq 1$ , find a packing of these items into minimum number of unit-size bins.

NP-hard.

$\alpha$ -approximation:  $\text{OPT} \leq \text{Alg.'s output} \leq \alpha \cdot \text{OPT}$

Even  $3/2$ -approximation is NP-hard: reduction from Partition  
Partition instance is scaled for total weight 2. If YES-instance, 2 bins are enough, otherwise 3 bins.

# Bin Packing Definition

## Definition (Bin Packing)

Given  $w_1, w_2, \dots, w_n$  such that  $0 < w_j \leq 1$ , find a packing of these items into minimum number of unit-size bins.

NP-hard.

$\alpha$ -approximation:  $\text{OPT} \leq \text{Alg.'s output} \leq \alpha \cdot \text{OPT} + c$

# Bin Packing Algorithms

- ▶ Constant approximation:  
First Fit ( $\frac{17}{10} \cdot \text{OPT} + 2$ ), Best Fit, ...  
First/Best Fit Decreasing ( $\frac{11}{9} \cdot \text{OPT} + 1$ ), ...
- ▶ Asymptotic PTAS:  $(1 + \epsilon) \cdot \text{OPT} + 1$  for any  $\epsilon > 0$   
[Fernandez de la Vega and Lueker 81]  
Linear in  $n$  and exponential in  $\frac{1}{\epsilon}$
- ▶ Better than PTAS:  $\text{OPT} + \log^2(\text{OPT})$   
[Karmarkar and Karp 82]  
 $O(n^8)$ -time algorithm
- ▶  $\text{OPT} + c$  possible? Open!

# Bin Packing Algorithms

- ▶ Constant approximation:  
First Fit ( $\frac{17}{10} \cdot \text{OPT} + 2$ ), Best Fit, ...  
First/Best Fit Decreasing ( $\frac{11}{9} \cdot \text{OPT} + 1$ ), ...
- ▶ Asymptotic PTAS:  $(1 + \epsilon) \cdot \text{OPT} + 1$  for any  $\epsilon > 0$   
[Fernandez de la Vega and Lueker 81]  
Linear in  $n$  and exponential in  $\frac{1}{\epsilon}$
- ▶ Better than PTAS:  $\text{OPT} + \log^2(\text{OPT})$   
[Karmarkar and Karp 82]  
 $O(n^8)$ -time algorithm
- ▶  $\text{OPT} + c$  possible? Open!

# Bin Packing Algorithms

- ▶ Constant approximation:  
First Fit ( $\frac{17}{10} \cdot \text{OPT} + 2$ ), Best Fit, ...  
First/Best Fit Decreasing ( $\frac{11}{9} \cdot \text{OPT} + 1$ ), ...
- ▶ Asymptotic PTAS:  $(1 + \epsilon) \cdot \text{OPT} + 1$  for any  $\epsilon > 0$   
[Fernandez de la Vega and Lueker 81]  
Linear in  $n$  and exponential in  $\frac{1}{\epsilon}$
- ▶ Better than PTAS:  $\text{OPT} + \log^2(\text{OPT})$   
[Karmarkar and Karp 82]  
 $O(n^8)$ -time algorithm
- ▶  $\text{OPT} + c$  possible? Open!

# Bin Packing Algorithms

- ▶ Constant approximation:  
First Fit ( $\frac{17}{10} \cdot \text{OPT} + 2$ ), Best Fit, ...  
First/Best Fit Decreasing ( $\frac{11}{9} \cdot \text{OPT} + 1$ ), ...
- ▶ Asymptotic PTAS:  $(1 + \epsilon) \cdot \text{OPT} + 1$  for any  $\epsilon > 0$   
[Fernandez de la Vega and Lueker 81]  
Linear in  $n$  and exponential in  $\frac{1}{\epsilon}$
- ▶ Better than PTAS:  $\text{OPT} + \log^2(\text{OPT})$   
[Karmarkar and Karp 82]  
 $O(n^8)$ -time algorithm
- ▶  $\text{OPT} + c$  possible? Open!

# Our Bin Packing Algorithm

We give a bin packing algorithm that

- ▶ outputs within  $(1 + \epsilon) \cdot \text{OPT} + 1$  for any  $\epsilon > 0$ ;
- ▶ runs in **sublinear-time** in  $n$ :  $o(n) + 2^{\text{poly}(\frac{1}{\epsilon})}$ ;
- ▶ has sampling access to the input.
- ▶ has error probability less than **0.01**.

# Input access

Input instance:  $w_1, w_2, \dots, w_n$

Uniform samples: Each  $(i, w_i)$  is sampled with prob.  $\frac{1}{n}$ .

Weighted samples: Each  $(i, w_i)$  is sampled with prob.  $\frac{w_i}{\sum_i w_i}$ .

# Lower Bounds: Uniform vs Weighted Samples

$\Omega(n)$  uniform samples are needed. Consider instances

$$\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}, 1, 1, 1 \quad \text{and} \quad \frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}$$

$\Omega(\sqrt{n})$  weighted samples are needed. Consider instances

$$\underbrace{1, 1, \dots, 1}_n \quad \text{and} \quad \underbrace{0, 0, \dots, 0}_{n/2}, \underbrace{1, 1, \dots, 1}_{n/2}$$

$\Omega(n^{1/3})$  weighted+uniform samples are needed. Consider instances

$$0, 0, \dots, 0, \underbrace{1, 1, \dots, 1}_{n^{2/3}} \quad \text{and} \quad 0, 0, \dots, 0, \underbrace{1, 1, \dots, 1}_{2n^{2/3}}$$

# Lower Bounds: Uniform vs Weighted Samples

$\Omega(n)$  uniform samples are needed. Consider instances

$$\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}, 1, 1, 1 \quad \text{and} \quad \frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}$$

$\Omega(\sqrt{n})$  weighted samples are needed. Consider instances

$$\underbrace{1, 1, \dots, 1}_n \quad \text{and} \quad \underbrace{0, 0, \dots, 0}_{n/2}, \underbrace{1, 1, \dots, 1}_{n/2}$$

$\Omega(n^{1/3})$  weighted+uniform samples are needed. Consider instances

$$0, 0, \dots, 0, \underbrace{1, 1, \dots, 1}_{n^{2/3}} \quad \text{and} \quad 0, 0, \dots, 0, \underbrace{1, 1, \dots, 1}_{2n^{2/3}}$$

# Lower Bounds: Uniform vs Weighted Samples

$\Omega(n)$  uniform samples are needed. Consider instances

$$\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}, 1, 1, 1 \quad \text{and} \quad \frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}$$

$\Omega(\sqrt{n})$  weighted samples are needed. Consider instances

$$\underbrace{1, 1, \dots, 1}_n \quad \text{and} \quad \underbrace{0, 0, \dots, 0}_{n/2}, \underbrace{1, 1, \dots, 1}_{n/2}$$

$\Omega(n^{1/3})$  weighted+uniform samples are needed. Consider instances

$$0, 0, \dots, 0, \underbrace{1, 1, \dots, 1}_{n^{2/3}} \quad \text{and} \quad 0, 0, \dots, 0, \underbrace{1, 1, \dots, 1}_{2n^{2/3}}$$

## 2-Approximation Algorithm

Let  $W = \sum_i w_i$ .

$$\lceil W \rceil \leq \text{OPT}$$

We can pack such that all bins are at least half full.

$$\text{OPT} \leq \lceil \frac{W}{\frac{1}{2}} \rceil = \lceil 2 \cdot W \rceil \leq 2 \cdot \lceil W \rceil \leq 2 \cdot \text{OPT}$$

Total weight  $W$  provides a simple 2-approximation.

# Estimating a Sum

## Uniform Samples

$\Omega(n)$  uniform samples are needed.  
Consider instance

$$\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}, 1, 1, 1$$

# Estimating a Sum

## Weighted Samples

$$B_0 = \left\{ i : w_i \leq \frac{\beta}{n} \right\}$$

# Estimating a Sum

## Weighted Samples

$$B_0 = \left\{ i : w_i \leq \frac{\beta}{n} \right\}$$

$$B_1 = \left\{ i : w_i \in \left( \frac{\beta}{n}, \frac{\beta \cdot (1 + \beta)}{n} \right] \right\}$$

# Estimating a Sum

## Weighted Samples

$$B_0 = \left\{ i : w_i \leq \frac{\beta}{n} \right\}$$

$$B_1 = \left\{ i : w_i \in \left( \frac{\beta}{n}, \frac{\beta \cdot (1 + \beta)}{n} \right] \right\}$$

$$B_2 = \left\{ i : w_i \in \left( \frac{\beta \cdot (1 + \beta)}{n}, \frac{\beta \cdot (1 + \beta)^2}{n} \right] \right\}$$

# Estimating a Sum

## Weighted Samples

$$B_0 = \left\{ i : w_i \leq \frac{\beta}{n} \right\}$$

$$B_j = \left\{ i : w_i \in \left( \frac{\beta \cdot (1 + \beta)^{j-1}}{n}, \frac{\beta \cdot (1 + \beta)^j}{n} \right] \right\} \text{ for } j = 1, \dots, \tilde{O}(\log n / \beta)$$

# Estimating a Sum

## Weighted Samples

$$B_0 = \left\{ i : w_i \leq \frac{\beta}{n} \right\}$$

$$B_j = \left\{ i : w_i \in \left( \frac{\beta \cdot (1 + \beta)^{j-1}}{n}, \frac{\beta \cdot (1 + \beta)^j}{n} \right] \right\} \text{ for } j = 1, \dots, \tilde{O}(\log n / \beta)$$

## Algorithm

1. Take  $\tilde{O}(\sqrt{n}/\epsilon^3)$  weighted samples.
2. Estimate  $|B_j|$  for those  $j$  with sufficient samples from  $B_j$ 
  - ▶ using collisions (= pairs of identical samples).
3. Estimate  $W$  using estimates for  $|B_j|$ 's.

# Cardinality Estimation from Samples

Suppose we get uniform samples from a  $k$ -element set (i.e., each with prob.  $\frac{1}{k}$ ). How do we determine  $k$ ?

## Algorithm

1. Take  $t$  samples until you see  $O(\frac{1}{\epsilon^2})$  collisions (pairs of identical samples).
2. Output  $\lceil O(\epsilon^2) \cdot \binom{t}{2} \rceil$ . ( $t = O(\sqrt{k})$ )

## For Weighted Samples

We need to apply a filtering since each  $i$  in  $B_j$  may have slightly different probability values.

# Cardinality Estimation from Samples

Suppose we get uniform samples from a  $k$ -element set (i.e., each with prob.  $\frac{1}{k}$ ). How do we determine  $k$ ?

## Algorithm

1. Take  $t$  samples until you see  $O(\frac{1}{\epsilon^2})$  collisions (pairs of identical samples).
2. Output  $\lceil O(\epsilon^2) \cdot \binom{t}{2} \rceil$ . ( $t = O(\sqrt{k})$ )

## For Weighted Samples

We need to apply a filtering since each  $i$  in  $B_j$  may have slightly different probability values.

# Cardinality Estimation from Samples

Suppose we get uniform samples from a  $k$ -element set (i.e., each with prob.  $\frac{1}{k}$ ). How do we determine  $k$ ?

## Algorithm

1. Take  $t$  samples until you see  $O(\frac{1}{\epsilon^2})$  collisions (pairs of identical samples).
2. Output  $\lceil O(\epsilon^2) \cdot \binom{t}{2} \rceil$ . ( $t = O(\sqrt{k})$ )

## For Weighted Samples

We need to apply a filtering since each  $i$  in  $B_j$  may have slightly different probability values.

# Estimating a Sum

## Weighted+Uniform Samples

$|B_i| \leq n^{2/3} \Rightarrow$  Use collisions:  $O(\sqrt{n^{2/3}}) = O(n^{1/3})$  wt. samples

$|B_i| > n^{2/3} \Rightarrow$  Use  $O(\frac{n}{n^{2/3}}) = O(n^{1/3})$  uniform samples

$\tilde{O}(n^{1/3}/\epsilon^3)$  weighted and uniform samples are sufficient to estimate the sum.

Sum estimation results are discovered simultaneously and independently by Motwani, Paningrahy, and Xu.

# Estimating a Sum

## Weighted+Uniform Samples

$|B_i| \leq n^{2/3} \Rightarrow$  Use collisions:  $O(\sqrt{n^{2/3}}) = O(n^{1/3})$  wt. samples

$|B_i| > n^{2/3} \Rightarrow$  Use  $O(\frac{n}{n^{2/3}}) = O(n^{1/3})$  uniform samples

$\tilde{O}(n^{1/3}/\epsilon^3)$  weighted and uniform samples are sufficient to estimate the sum.

Sum estimation results are discovered simultaneously and independently by Motwani, Paningrahy, and Xu.

# Estimating a Sum

## Weighted+Uniform Samples

$|B_i| \leq n^{2/3} \Rightarrow$  Use collisions:  $O(\sqrt{n^{2/3}}) = O(n^{1/3})$  wt. samples

$|B_i| > n^{2/3} \Rightarrow$  Use  $O(\frac{n}{n^{2/3}}) = O(n^{1/3})$  uniform samples

$\tilde{O}(n^{1/3}/\epsilon^3)$  weighted and uniform samples are sufficient to estimate the sum.

Sum estimation results are discovered simultaneously and independently by Motwani, Paningrahy, and Xu.

# Bin Packing Algorithm

## Our bin packing algorithm

- ▶ outputs within  $(1 + \epsilon) \cdot \text{OPT} + 1$  for any  $\epsilon > 0$ ;
- ▶ uses  $\tilde{O}(\sqrt{n}/\epsilon^5)$  weighted or  $\tilde{O}(n^{1/3}/\epsilon^5)$  weighted+uniform samples;
- ▶ runs in time linear in the number of samples +  $2^{\text{poly}(\frac{1}{\epsilon})}$ .
- ▶ has error probability less than 0.01.

## Special Case: Small Items

Suppose  $w_i \leq \beta$  for all  $i$ .

Each bin can be filled up to  $1 - \beta$  level. Hence,

$$\lceil \frac{W}{1 - \beta} \rceil \leq \lceil (1 + 2\beta) \cdot W \rceil \leq (1 + 2\beta) \cdot \lceil W \rceil + 1 \leq (1 + 2\beta) \cdot \text{OPT} + 1$$

Estimating  $W$  is sufficient for a  $(1 + \epsilon)$ -approximation.

## Special Case: Large Items

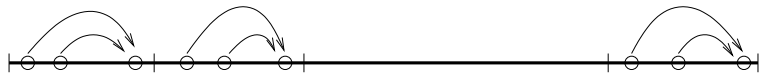
If  $w_i > \beta$  for all  $i$ , each bin contains at most  $\lfloor \frac{1}{\beta} \rfloor$  items.

### Lemma

*If the number of different weights in  $w_1, \dots, w_n$  is a constant  $c$  and  $w_i \geq \beta$  for all  $i$ , bin packing can be solved **exactly** in time  $O(n^{c/\beta})$ .*

Proof idea: There are  $O(n^c)$  different subsets of items and  $O(n^{1/\beta})$  configurations for a bin.

Dynamic Programming table of size  $O(n^c)$  with time  $O(n^{1/\beta})$  to fill in each table entry.

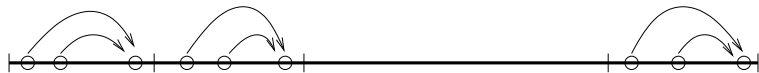


$\beta^2 \cdot n$  items

## Algorithm

1. Group the items into constant number of groups (in nondecreasing order).
2. Round up the weight of each item to largest weight in the group (constant number of different weights).
3. Using previous lemma, output the optimal packing of these items.

**Note.** This idea still works if we know the weights of items that are near the maximum of each group. Sampling can find such representatives.



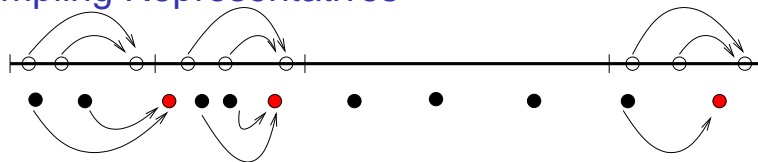
$\beta^2 \cdot n$  items

## Algorithm

1. Group the items into constant number of groups (in nondecreasing order).
2. Round up the weight of each item to largest weight in the group (constant number of different weights).
3. Using previous lemma, output the optimal packing of these items.

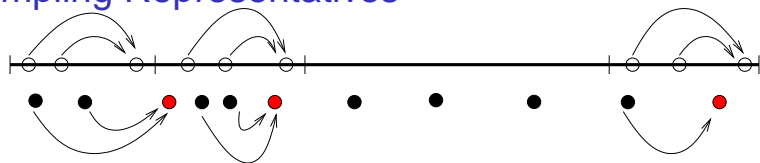
**Note.** This idea still works if we know the weights of items that are near the maximum of each group. Sampling can find such representatives.

# Sampling Representatives



Since there are constant number of groups, sample size is independent of  $n$ .

# Sampling Representatives



Since there are constant number of groups, sample size is independent of  $n$ .

**Sublinear?** Scale down the numbers of items in each group to a constant  $k(\beta)$  (exponential in  $\frac{1}{\beta}$ ). Solve scaled-down instance. Scale solution back up.

$$\beta^2 \cdot n \times v_1, \dots, \beta^2 \cdot n \times v_{\beta-2}$$

to

$$k(\beta) \times v_1, \dots, k(\beta) \times v_{\beta-2}$$

# General Bin Packing Algorithm

## Algorithm

1. Small item  $i$ :  $w_i < \epsilon/4$ .
2. Obtain a  $(1 + \epsilon/4)$ -approximation  $\tilde{W}$  to  $W$ .
3. Obtain a  $(1 + \epsilon/4)$ -approximation  $b$  to cost of optimal packing of large items (unless large items are insignificant).
4. Output  $\lceil (1 + \epsilon/2) \cdot \max\{\tilde{W}, b\} \rceil$ .

# General Bin Packing Algorithm

## Algorithm

1. Small item  $i$ :  $w_i < \epsilon/4$ .
2. Obtain a  $(1 + \epsilon/4)$ -approximation  $\tilde{W}$  to  $W$ .
3. Obtain a  $(1 + \epsilon/4)$ -approximation  $b$  to cost of optimal packing of large items (unless large items are insignificant).
4. Output  $\lceil (1 + \epsilon/2) \cdot \max\{\tilde{W}, b\} \rceil$ .

# General Bin Packing Algorithm

## Algorithm

1. Small item  $i$ :  $w_i < \epsilon/4$ .
2. Obtain a  $(1 + \epsilon/4)$ -approximation  $\tilde{W}$  to  $W$ .
3. Obtain a  $(1 + \epsilon/4)$ -approximation  $b$  to cost of optimal packing of large items (unless large items are insignificant).
4. Output  $\lceil (1 + \epsilon/2) \cdot \max\{\tilde{W}, b\} \rceil$ .

# General Bin Packing Algorithm

## Algorithm

1. Small item  $i$ :  $w_i < \epsilon/4$ .
2. Obtain a  $(1 + \epsilon/4)$ -approximation  $\tilde{W}$  to  $W$ .
3. Obtain a  $(1 + \epsilon/4)$ -approximation  $b$  to cost of optimal packing of large items (unless large items are insignificant).
4. Output  $\lceil (1 + \epsilon/2) \cdot \max\{\tilde{W}, b\} \rceil$ .

# General Bin Packing Algorithm

## Algorithm

1. Small item  $i$ :  $w_i < \epsilon/4$ .
2. Obtain a  $(1 + \epsilon/4)$ -approximation  $\tilde{W}$  to  $W$ .
3. Obtain a  $(1 + \epsilon/4)$ -approximation  $b$  to cost of optimal packing of large items (unless large items are insignificant).
4. Output  $\lceil (1 + \epsilon/2) \cdot \max\{\tilde{W}, b\} \rceil$ .

If  $\text{OPT} = \text{OPT}_H$ ,

$$\text{OPT} = \text{OPT}_H \leq \lceil (1 + \epsilon/2) \cdot \max\{\tilde{W}, b\} \rceil.$$

Otherwise,

$$\text{OPT} \leq \lceil \frac{W}{1 - \epsilon/4} \rceil \leq \lceil (1 + \epsilon/2) \cdot \max\{\tilde{W}, b\} \rceil.$$

# General Bin Packing Algorithm

## Algorithm

1. Small item  $i$ :  $w_i < \epsilon/4$ .
2. Obtain a  $(1 + \epsilon/4)$ -approximation  $\tilde{W}$  to  $W$ .
3. Obtain a  $(1 + \epsilon/4)$ -approximation  $b$  to cost of optimal packing of large items (unless large items are insignificant).
4. Output  $\lceil (1 + \epsilon/2) \cdot \max\{\tilde{W}, b\} \rceil$ .

$$\begin{aligned}\lceil (1 + \epsilon/2) \cdot \max\{\tilde{W}, b\} \rceil &\leq \lceil (1 + \epsilon/2) \cdot (1 + \epsilon/4) \cdot \max\{W, \text{OPT}_H\} \rceil \\ &\leq (1 + \epsilon) \cdot \lceil \max\{W, \text{OPT}_H\} \rceil + 1 \\ &\leq (1 + \epsilon) \cdot \text{OPT} + 1\end{aligned}$$

# To Sum Up

We gave a sublinear-time approximation scheme for bin packing problem.

- ▶ Weighted samples were necessary.
- ▶ Additional uniform samples improved the complexity.
- ▶ Tight in terms of dependency on  $n$ .

## Remark 1.

The algorithm can output a constant-size “template” packing to help obtain an approximate solution.

## Remark 2.

If  $W$  (or an approximation to  $W$ ) is given as part of input, sample (and time) complexity is independent of  $n$ .

# To Sum Up

We gave a sublinear-time approximation scheme for bin packing problem.

- ▶ Weighted samples were necessary.
- ▶ Additional uniform samples improved the complexity.
- ▶ Tight in terms of dependency on  $n$ .

## Remark 1.

The algorithm can output a constant-size “template” packing to help obtain an approximate solution.

## Remark 2.

If  $W$  (or an approximation to  $W$ ) is given as part of input, sample (and time) complexity is independent of  $n$ .

# To Sum Up

We gave a sublinear-time approximation scheme for bin packing problem.

- ▶ Weighted samples were necessary.
- ▶ Additional uniform samples improved the complexity.
- ▶ Tight in terms of dependency on  $n$ .

## Remark 1.

The algorithm can output a constant-size “template” packing to help obtain an approximate solution.

## Remark 2.

If  $W$  (or an approximation to  $W$ ) is given as part of input, sample (and time) complexity is independent of  $n$ .

## Question(s).

1. What other combinatorial optimization problems yield themselves to sublinear algorithms?
2. What extra information could be provided to reduce complexity?