Asymptotic PTAS for Bin-Packing

Vahan Mkrtchyan[1]

[1] Lane Department of Computer Science and Electrical Engineering
West Virginia University

March 21, 2014

Outline

Outline

Outline

Outline

Outline

Outline

Topics

Topics

Outline

Topics

Outline

1. Problem definition.

Topics

Outline

1. Problem definition.
2. Definition of PTAS, FPTAS and Asymptotic PTAS.

Topics

Outline

1. Problem definition.
2. Definition of PTAS, FPTAS and Asymptotic PTAS.
3. Polynomial algorithm for restricted instances.

Topics

Outline

1. Problem definition.
2. Definition of PTAS, FPTAS and Asymptotic PTAS.
3. Polynomial algorithm for restricted instances.
4. Approximation algorithm for restricted instances.

Topics

Outline

1. Problem definition.
2. Definition of PTAS, FPTAS and Asymptotic PTAS.
3. Polynomial algorithm for restricted instances.
4. Approximation algorithm for restricted instances.
5. Asymptotic PTAS for Bin Packing.

Problem definition

Problem Statement

We are given n objects of sizes $\{s_1, s_2, \ldots s_n\}$, such that $0 < s_i \leq 1$ and an unlimited supply of unit sized bins.

Problem definition

Problem Statement

We are given n objects of sizes $\{s_1, s_2, \ldots s_n\}$, such that $0 < s_i \leq 1$ and an unlimited supply of unit sized bins. The goal is to pack the objects into bins, minimizing the number of bins used.

PTAS, FPTAS and Asymptotic PTAS

Definition

A PTAS for a minimization problem Π is an algorithm A, which on all instances I of Π and error-parameter $\varepsilon > 0$, returns a solution of cost $A(I)$, such that $A(I) \leq (1+\varepsilon) \cdot OPT$.

PTAS, FPTAS and Asymptotic PTAS

Definition

A PTAS for a minimization problem Π is an algorithm A, which on all instances I of Π and error-parameter $\varepsilon > 0$, returns a solution of cost $A(I)$, such that $A(I) \leq (1+\varepsilon) \cdot OPT$. The running time of the algorithm must be polynomial for each fixed value of ε.

PTAS, FPTAS and Asymptotic PTAS

Definition

A PTAS for a minimization problem Π is an algorithm A, which on all instances I of Π and error-parameter $\varepsilon > 0$, returns a solution of cost $A(I)$, such that $A(I) \leq (1+\varepsilon) \cdot OPT$. The running time of the algorithm must be polynomial for each fixed value of ε.

Definition

An FPTAS for a minimization problem Π is a PTAS that runs in time, that is polynomial from the size of the input instance and $\frac{1}{\varepsilon}$.

PTAS, FPTAS and Asymptotic PTAS

Definition

A PTAS for a minimization problem Π is an algorithm A, which on all instances I of Π and error-parameter $\varepsilon > 0$, returns a solution of cost $A(I)$, such that $A(I) \leq (1+\varepsilon) \cdot OPT$. The running time of the algorithm must be polynomial for each fixed value of ε.

Definition

An FPTAS for a minimization problem Π is a PTAS that runs in time, that is polynomial from the size of the input instance and $\frac{1}{\varepsilon}$.

Definition

An asymptotic PTAS for a minimization problem Π is an algorithm A, which on all instances I of Π and error-parameter $\varepsilon > 0$, returns a solution of cost $A(I)$, such that
$A(I) \leq (1+\varepsilon) \cdot OPT + C(\varepsilon)$.

PTAS, FPTAS and Asymptotic PTAS

Definition

A PTAS for a minimization problem Π is an algorithm A, which on all instances I of Π and error-parameter $\varepsilon > 0$, returns a solution of cost $A(I)$, such that $A(I) \leq (1+\varepsilon) \cdot OPT$. The running time of the algorithm must be polynomial for each fixed value of ε.

Definition

An FPTAS for a minimization problem Π is a PTAS that runs in time, that is polynomial from the size of the input instance and $\frac{1}{\varepsilon}$.

Definition

An asymptotic PTAS for a minimization problem Π is an algorithm A, which on all instances I of Π and error-parameter $\varepsilon > 0$, returns a solution of cost $A(I)$, such that $A(I) \leq (1+\varepsilon) \cdot OPT + C(\varepsilon)$. The running time of the algorithm must be polynomial for each fixed value of ε.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$, and the number of different item-sizes is K.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$, and the number of different item-sizes is K. Then there is a polynomial algorithm for solving these instances.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$, and the number of different item-sizes is K. Then there is a polynomial algorithm for solving these instances.

Proof.

(Sketch). We will show that the number of feasible packings is at most polynomial.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$, and the number of different item-sizes is K. Then there is a polynomial algorithm for solving these instances.

Proof.

(Sketch). We will show that the number of feasible packings is at most polynomial.

(1) The number of items in a bin is at most $M = \lfloor \frac{1}{\varepsilon} \rfloor$.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$, and the number of different item-sizes is K. Then there is a polynomial algorithm for solving these instances.

Proof.

(Sketch). We will show that the number of feasible packings is at most polynomial.

(1) The number of items in a bin is at most $M = \lfloor \frac{1}{\varepsilon} \rfloor$.

(2) The number of different bin-types is at most $R = \binom{\tilde{K}}{M} = \binom{K+M-1}{M}$.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$, and the number of different item-sizes is K. Then there is a polynomial algorithm for solving these instances.

Proof.

(Sketch). We will show that the number of feasible packings is at most polynomial.

(1) The number of items in a bin is at most $M = \lfloor \frac{1}{\varepsilon} \rfloor$.

(2) The number of different bin-types is at most $R = \binom{\tilde{K}}{M} = \binom{K+M-1}{M}$.

(3) No more than n bins are needed in a feasible solution of any instance.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$, and the number of different item-sizes is K. Then there is a polynomial algorithm for solving these instances.

Proof.

(Sketch). We will show that the number of feasible packings is at most polynomial.

(1) The number of items in a bin is at most $M = \lfloor \frac{1}{\varepsilon} \rfloor$.

(2) The number of different bin-types is at most $R = \binom{\tilde{K}}{M} = \binom{K+M-1}{M}$.

(3) No more than n bins are needed in a feasible solution of any instance.

(4) The number of feasible packings is at most $\binom{\tilde{R}}{n} = \binom{R+n-1}{n} = \binom{R+n-1}{R-1} = O(n^{R-1})$.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$, and the number of different item-sizes is K. Then there is a polynomial algorithm for solving these instances.

Proof.

(Sketch). We will show that the number of feasible packings is at most polynomial.

(1) The number of items in a bin is at most $M = \lfloor \frac{1}{\varepsilon} \rfloor$.

(2) The number of different bin-types is at most $R = \binom{\tilde{K}}{M} = \binom{K+M-1}{M}$.

(3) No more than n bins are needed in a feasible solution of any instance.

(4) The number of feasible packings is at most $\binom{\tilde{R}}{n} = \binom{R+n-1}{n} = \binom{R+n-1}{R-1} = O(n^{R-1})$.

Clearly, we can go through all of them, and find the one that uses minimum number of bins. □

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$. *Then there is a* $(1+\varepsilon)$*-approximation algorithm for solving these instances.*

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$. Then there is a $(1+\varepsilon)$-approximation algorithm for solving these instances.

The Algorithm

- Let $K = \lceil \frac{1}{\varepsilon^2} \rceil$ and $Q = \lfloor n \cdot \varepsilon^2 \rfloor$.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$. Then there is a $(1+\varepsilon)$-approximation algorithm for solving these instances.

The Algorithm

- Let $K = \lceil \frac{1}{\varepsilon^2} \rceil$ and $Q = \lfloor n \cdot \varepsilon^2 \rfloor$.
- Sort the sizes of the items of the input instance I as follows: $s_1 \leq s_2 \leq \ldots \leq s_n$.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$. Then there is a $(1+\varepsilon)$-approximation algorithm for solving these instances.

The Algorithm

- Let $K = \lceil \frac{1}{\varepsilon^2} \rceil$ and $Q = \lfloor n \cdot \varepsilon^2 \rfloor$.
- Sort the sizes of the items of the input instance I as follows: $s_1 \leq s_2 \leq \ldots \leq s_n$.
- Partition the items into K groups, each of which is of size Q (except may be the last one).

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$. Then there is a $(1+\varepsilon)$-approximation algorithm for solving these instances.

The Algorithm

- Let $K = \lceil \frac{1}{\varepsilon^2} \rceil$ and $Q = \lfloor n \cdot \varepsilon^2 \rfloor$.
- Sort the sizes of the items of the input instance I as follows: $s_1 \leq s_2 \leq \ldots \leq s_n$.
- Partition the items into K groups, each of which is of size Q (except may be the last one).
- Consider the new instance J of bin packing obtained from I by rounding the size of each element to the maximum size of its group.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$. Then there is a $(1+\varepsilon)$-approximation algorithm for solving these instances.

The Algorithm

- Let $K = \lceil \frac{1}{\varepsilon^2} \rceil$ and $Q = \lfloor n \cdot \varepsilon^2 \rfloor$.
- Sort the sizes of the items of the input instance I as follows: $s_1 \leq s_2 \leq \ldots \leq s_n$.
- Partition the items into K groups, each of which is of size Q (except may be the last one).
- Consider the new instance J of bin packing obtained from I by rounding the size of each element to the maximum size of its group.
- Solve the instance J by previous algorithm.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$. Then there is a $(1+\varepsilon)$-approximation algorithm for solving these instances.

The Algorithm

- Let $K = \lceil \frac{1}{\varepsilon^2} \rceil$ and $Q = \lfloor n \cdot \varepsilon^2 \rfloor$.
- Sort the sizes of the items of the input instance I as follows: $s_1 \leq s_2 \leq \ldots \leq s_n$.
- Partition the items into K groups, each of which is of size Q (except may be the last one).
- Consider the new instance J of bin packing obtained from I by rounding the size of each element to the maximum size of its group.
- Solve the instance J by previous algorithm.
- Return the packing of J as a packing of I.

Restricted Instances

Theorem

Consider the instances of bin packing, in which the sizes of items are $\geq \varepsilon$. Then there is a $(1+\varepsilon)$-approximation algorithm for solving these instances.

The Algorithm

- Let $K = \lceil \frac{1}{\varepsilon^2} \rceil$ and $Q = \lfloor n \cdot \varepsilon^2 \rfloor$.
- Sort the sizes of the items of the input instance I as follows: $s_1 \leq s_2 \leq \ldots \leq s_n$.
- Partition the items into K groups, each of which is of size Q (except may be the last one).
- Consider the new instance J of bin packing obtained from I by rounding the size of each element to the maximum size of its group.
- Solve the instance J by previous algorithm.
- Return the packing of J as a packing of I.

Remark

Observe that the packing returned by the algorithm is a feasible packing of I.

Restricted Instances

Some Instances

- I-the input instance.

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.
- J_Q- the instance obtained from J by removing the last Q items.

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.
- J_Q- the instance obtained from J by removing the last Q items.
- J'_Q- the instance obtained from J' by removing the first Q items.

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.
- J_Q- the instance obtained from J by removing the last Q items.
- J'_Q- the instance obtained from J' by removing the first Q items.

The Analysis

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.
- J_Q- the instance obtained from J by removing the last Q items.
- J'_Q- the instance obtained from J' by removing the first Q items.

The Analysis

$$OPT(J) \quad \leq$$

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.
- J_Q- the instance obtained from J by removing the last Q items.
- J'_Q- the instance obtained from J' by removing the first Q items.

The Analysis

$$OPT(J) \quad \leq \quad OPT(J_Q) + Q$$

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.
- J_Q- the instance obtained from J by removing the last Q items.
- J'_Q- the instance obtained from J' by removing the first Q items.

The Analysis

$$\begin{aligned} OPT(J) &\leq OPT(J_Q) + Q \\ &\leq \end{aligned}$$

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.
- J_Q- the instance obtained from J by removing the last Q items.
- J'_Q- the instance obtained from J' by removing the first Q items.

The Analysis

$$\begin{aligned} OPT(J) &\leq OPT(J_Q) + Q \\ &\leq OPT(J'_Q) + Q \end{aligned}$$

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.
- J_Q- the instance obtained from J by removing the last Q items.
- J'_Q- the instance obtained from J' by removing the first Q items.

The Analysis

$$\begin{aligned} OPT(J) &\leq OPT(J_Q) + Q \\ &\leq OPT(J'_Q) + Q \\ &\leq \end{aligned}$$

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.
- J_Q- the instance obtained from J by removing the last Q items.
- J'_Q- the instance obtained from J' by removing the first Q items.

The Analysis

$$\begin{aligned} OPT(J) &\leq OPT(J_Q) + Q \\ &\leq OPT(J'_Q) + Q \\ &\leq OPT(J') + Q \end{aligned}$$

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.
- J_Q- the instance obtained from J by removing the last Q items.
- J'_Q- the instance obtained from J' by removing the first Q items.

The Analysis

$$\begin{aligned} OPT(J) &\leq OPT(J_Q) + Q \\ &\leq OPT(J'_Q) + Q \\ &\leq OPT(J') + Q \\ &\leq \end{aligned}$$

Restricted Instances

Some Instances

- I-the input instance.
- J- the instance constructed by the algorithm.
- J'- the instance obtained from I by rounding the size of each element to the minimum size of its group.
- J_Q- the instance obtained from J by removing the last Q items.
- J'_Q- the instance obtained from J' by removing the first Q items.

The Analysis

$$\begin{aligned} OPT(J) &\leq OPT(J_Q) + Q \\ &\leq OPT(J'_Q) + Q \\ &\leq OPT(J') + Q \\ &\leq OPT(I) + Q. \end{aligned}$$

Restricted Instances

Some Bounds

$$Q =$$

Restricted Instances

Some Bounds

$$Q = \lfloor n \cdot \varepsilon^2 \rfloor$$

Restricted Instances

Some Bounds

$$\begin{aligned} Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\ &\leq \end{aligned}$$

Restricted Instances

Some Bounds

$$
\begin{aligned}
Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\
&\leq n \cdot \varepsilon^2
\end{aligned}
$$

Restricted Instances

Some Bounds

$$
\begin{aligned}
Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\
&\leq n \cdot \varepsilon^2 \\
&\leq
\end{aligned}
$$

Restricted Instances

Some Bounds

$$
\begin{aligned}
Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\
&\leq n \cdot \varepsilon^2 \\
&\leq \varepsilon \cdot \sum_{i=1}^{n} s_i
\end{aligned}
$$

Restricted Instances

Some Bounds

$$
\begin{aligned}
Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\
&\leq n \cdot \varepsilon^2 \\
&\leq \varepsilon \cdot \sum_{i=1}^{n} s_i \\
&\leq
\end{aligned}
$$

Restricted Instances

Some Bounds

$$
\begin{aligned}
Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\
&\leq n \cdot \varepsilon^2 \\
&\leq \varepsilon \cdot \sum_{i=1}^{n} s_i \\
&\leq \varepsilon \cdot OPT.
\end{aligned}
$$

Restricted Instances

Some Bounds

$$\begin{aligned} Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\ &\leq n \cdot \varepsilon^2 \\ &\leq \varepsilon \cdot \sum_{i=1}^{n} s_i \\ &\leq \varepsilon \cdot OPT. \end{aligned}$$

Therefore:

$$OPT(J) \quad \leq$$

Restricted Instances

Some Bounds

$$\begin{aligned} Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\ &\leq n \cdot \varepsilon^2 \\ &\leq \varepsilon \cdot \sum_{i=1}^{n} s_i \\ &\leq \varepsilon \cdot OPT. \end{aligned}$$

Therefore:

$$OPT(J) \leq OPT(I) + Q$$

Restricted Instances

Some Bounds

$$\begin{aligned} Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\ &\leq n \cdot \varepsilon^2 \\ &\leq \varepsilon \cdot \sum_{i=1}^{n} s_i \\ &\leq \varepsilon \cdot OPT. \end{aligned}$$

Therefore:

$$\begin{aligned} OPT(J) &\leq OPT(I) + Q \\ &\leq \end{aligned}$$

Restricted Instances

Some Bounds

$$
\begin{aligned}
Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\
&\leq n \cdot \varepsilon^2 \\
&\leq \varepsilon \cdot \sum_{i=1}^{n} s_i \\
&\leq \varepsilon \cdot OPT.
\end{aligned}
$$

Therefore:

$$
\begin{aligned}
OPT(J) &\leq OPT(I) + Q \\
&\leq OPT(I) + \varepsilon \cdot OPT
\end{aligned}
$$

Restricted Instances

Some Bounds

$$\begin{aligned} Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\ &\leq n \cdot \varepsilon^2 \\ &\leq \varepsilon \cdot \sum_{i=1}^{n} s_i \\ &\leq \varepsilon \cdot OPT. \end{aligned}$$

Therefore:

$$\begin{aligned} OPT(J) &\leq OPT(I) + Q \\ &\leq OPT(I) + \varepsilon \cdot OPT \\ &= \end{aligned}$$

Restricted Instances

Some Bounds

$$\begin{aligned} Q &= \lfloor n \cdot \varepsilon^2 \rfloor \\ &\leq n \cdot \varepsilon^2 \\ &\leq \varepsilon \cdot \sum_{i=1}^{n} s_i \\ &\leq \varepsilon \cdot OPT. \end{aligned}$$

Therefore:

$$\begin{aligned} OPT(J) &\leq OPT(I) + Q \\ &\leq OPT(I) + \varepsilon \cdot OPT \\ &= (1+\varepsilon) \cdot OPT. \end{aligned}$$

The General Case

Theorem

There exists a polynomial algorithm that for each $\varepsilon \in (0, \frac{1}{2}]$ *finds a packing with the number of bins at most* $(1 + 2 \cdot \varepsilon) \cdot OPT + 1$.

The General Case

Theorem

There exists a polynomial algorithm that for each $\varepsilon \in (0, \frac{1}{2}]$ *finds a packing with the number of bins at most* $(1 + 2 \cdot \varepsilon) \cdot OPT + 1$*. In other words, bin packing problem admits an asymptotic PTAS.*

The General Case

Theorem

There exists a polynomial algorithm that for each $\varepsilon \in (0, \frac{1}{2}]$ *finds a packing with the number of bins at most* $(1 + 2 \cdot \varepsilon) \cdot OPT + 1$*. In other words, bin packing problem admits an asymptotic PTAS.*

The Algorithm

- For the input instance I consider the instance I' obtained from I by removing all items of size less than ε.

The General Case

Theorem

There exists a polynomial algorithm that for each $\varepsilon \in (0, \frac{1}{2}]$ finds a packing with the number of bins at most $(1 + 2 \cdot \varepsilon) \cdot OPT + 1$. In other words, bin packing problem admits an asymptotic PTAS.

The Algorithm

- For the input instance I consider the instance I' obtained from I by removing all items of size less than ε.
- Solve I' by the previous $(1 + \varepsilon)$-approximation algorithm.

The General Case

Theorem

There exists a polynomial algorithm that for each $\varepsilon \in (0, \frac{1}{2}]$ finds a packing with the number of bins at most $(1 + 2 \cdot \varepsilon) \cdot OPT + 1$. In other words, bin packing problem admits an asymptotic PTAS.

The Algorithm

- For the input instance I consider the instance I' obtained from I by removing all items of size less than ε.
- Solve I' by the previous $(1 + \varepsilon)$-approximation algorithm.
- Apply First-Fit on the resulting packing using the items from $I - I'$.

The General Case

Theorem

There exists a polynomial algorithm that for each $\varepsilon \in (0, \frac{1}{2}]$ finds a packing with the number of bins at most $(1 + 2 \cdot \varepsilon) \cdot OPT + 1$. In other words, bin packing problem admits an asymptotic PTAS.

The Algorithm

- For the input instance I consider the instance I' obtained from I by removing all items of size less than ε.
- Solve I' by the previous $(1 + \varepsilon)$-approximation algorithm.
- Apply First-Fit on the resulting packing using the items from $I - I'$.
- Return the resulting packing.

The Analysis of the Algorithm

The Analysis

Let L be the number of bins returned by the algorithm.

The Analysis of the Algorithm

The Analysis

Let L be the number of bins returned by the algorithm. If no extra bin was required for packing the items from $I - I'$, then $L \leq (1+\varepsilon) \cdot OPT(I') \leq (1+\varepsilon) \cdot OPT(I)$,

The Analysis of the Algorithm

The Analysis

Let L be the number of bins returned by the algorithm. If no extra bin was required for packing the items from $I - I'$, then $L \leq (1+\varepsilon) \cdot OPT(I') \leq (1+\varepsilon) \cdot OPT(I)$, hence we can assume that extra bins were required.

The Analysis of the Algorithm

The Analysis

Let L be the number of bins returned by the algorithm. If no extra bin was required for packing the items from $I - I'$, then $L \leq (1+\varepsilon) \cdot OPT(I') \leq (1+\varepsilon) \cdot OPT(I)$, hence we can assume that extra bins were required.

The Analysis: Extra Bins Were Required

The room in the first $L-1$ bins is less than ε.

The Analysis of the Algorithm

The Analysis

Let L be the number of bins returned by the algorithm. If no extra bin was required for packing the items from $I - I'$, then $L \leq (1+\varepsilon) \cdot OPT(I') \leq (1+\varepsilon) \cdot OPT(I)$, hence we can assume that extra bins were required.

The Analysis: Extra Bins Were Required

The room in the first $L-1$ bins is less than ε. Then:

$$OPT \quad \geq$$

The Analysis of the Algorithm

The Analysis

Let L be the number of bins returned by the algorithm. If no extra bin was required for packing the items from $I - I'$, then $L \leq (1+\varepsilon) \cdot OPT(I') \leq (1+\varepsilon) \cdot OPT(I)$, hence we can assume that extra bins were required.

The Analysis: Extra Bins Were Required

The room in the first $L-1$ bins is less than ε. Then:

$$OPT \geq \sum_{i=1}^{n} s_i$$

The Analysis of the Algorithm

The Analysis

Let L be the number of bins returned by the algorithm. If no extra bin was required for packing the items from $I - I'$, then $L \leq (1+\varepsilon) \cdot OPT(I') \leq (1+\varepsilon) \cdot OPT(I)$, hence we can assume that extra bins were required.

The Analysis: Extra Bins Were Required

The room in the first $L-1$ bins is less than ε. Then:

$$\begin{aligned} OPT &\geq \sum_{i=1}^{n} s_i \\ &> \end{aligned}$$

The Analysis of the Algorithm

The Analysis

Let L be the number of bins returned by the algorithm. If no extra bin was required for packing the items from $I - I'$, then $L \leq (1+\varepsilon) \cdot OPT(I') \leq (1+\varepsilon) \cdot OPT(I)$, hence we can assume that extra bins were required.

The Analysis: Extra Bins Were Required

The room in the first $L-1$ bins is less than ε. Then:

$$\begin{aligned} OPT &\geq \sum_{i=1}^{n} s_i \\ &> (L-1) \cdot (1-\varepsilon) \end{aligned}$$

The Analysis of the Algorithm

The Analysis

Let L be the number of bins returned by the algorithm. If no extra bin was required for packing the items from $I - I'$, then $L \leq (1+\varepsilon) \cdot OPT(I') \leq (1+\varepsilon) \cdot OPT(I)$, hence we can assume that extra bins were required.

The Analysis: Extra Bins Were Required

The room in the first $L - 1$ bins is less than ε. Then:

$$\begin{aligned} OPT &\geq \sum_{i=1}^{n} s_i \\ &> (L-1)\cdot(1-\varepsilon) \end{aligned}$$

or

$$L < \frac{OPT}{1-\varepsilon} + 1.$$

The Analysis of the Algorithm

The Final Bound

We need to show that $\frac{OPT}{1-\varepsilon} + 1 \leq (1 + 2 \cdot \varepsilon) \cdot OPT + 1$.

The Analysis of the Algorithm

The Final Bound

We need to show that $\frac{OPT}{1-\varepsilon} + 1 \leq (1 + 2 \cdot \varepsilon) \cdot OPT + 1$.

$$1 + 2 \cdot \varepsilon - \frac{1}{1-\varepsilon} \quad =$$

The Analysis of the Algorithm

The Final Bound

We need to show that $\frac{OPT}{1-\varepsilon} + 1 \leq (1 + 2 \cdot \varepsilon) \cdot OPT + 1$.

$$1 + 2 \cdot \varepsilon - \frac{1}{1-\varepsilon} \quad = \quad \frac{1}{1-\varepsilon} \cdot ((1-\varepsilon)(1 + 2 \cdot \varepsilon) - 1)$$

The Analysis of the Algorithm

The Final Bound

We need to show that $\frac{OPT}{1-\varepsilon} + 1 \leq (1 + 2 \cdot \varepsilon) \cdot OPT + 1$.

$$
\begin{aligned}
1 + 2 \cdot \varepsilon - \frac{1}{1-\varepsilon} &= \frac{1}{1-\varepsilon} \cdot ((1-\varepsilon)(1+2\cdot\varepsilon) - 1) \\
&=
\end{aligned}
$$

The Analysis of the Algorithm

The Final Bound

We need to show that $\frac{OPT}{1-\varepsilon}+1 \leq (1+2\cdot\varepsilon)\cdot OPT+1$.

$$\begin{aligned} 1+2\cdot\varepsilon-\frac{1}{1-\varepsilon} &= \frac{1}{1-\varepsilon}\cdot((1-\varepsilon)(1+2\cdot\varepsilon)-1) \\ &= \frac{1}{1-\varepsilon}\cdot(1+\varepsilon-2\cdot\varepsilon^2-1) \end{aligned}$$

The Analysis of the Algorithm

The Final Bound

We need to show that $\frac{OPT}{1-\varepsilon} + 1 \leq (1 + 2 \cdot \varepsilon) \cdot OPT + 1$.

$$\begin{aligned} 1 + 2 \cdot \varepsilon - \frac{1}{1-\varepsilon} &= \frac{1}{1-\varepsilon} \cdot ((1-\varepsilon)(1+2\cdot\varepsilon) - 1) \\ &= \frac{1}{1-\varepsilon} \cdot (1 + \varepsilon - 2 \cdot \varepsilon^2 - 1) \\ &= \end{aligned}$$

The Analysis of the Algorithm

The Final Bound

We need to show that $\frac{OPT}{1-\varepsilon}+1 \leq (1+2\cdot\varepsilon)\cdot OPT+1$.

$$\begin{aligned}
1+2\cdot\varepsilon-\frac{1}{1-\varepsilon} &= \frac{1}{1-\varepsilon}\cdot((1-\varepsilon)(1+2\cdot\varepsilon)-1) \\
&= \frac{1}{1-\varepsilon}\cdot(1+\varepsilon-2\cdot\varepsilon^2-1) \\
&= \frac{\varepsilon}{1-\varepsilon}\cdot(1-2\cdot\varepsilon)
\end{aligned}$$

The Analysis of the Algorithm

The Final Bound

We need to show that $\frac{OPT}{1-\varepsilon} + 1 \leq (1 + 2 \cdot \varepsilon) \cdot OPT + 1$.

$$\begin{aligned} 1 + 2 \cdot \varepsilon - \frac{1}{1-\varepsilon} &= \frac{1}{1-\varepsilon} \cdot ((1-\varepsilon)(1+2\cdot\varepsilon) - 1) \\ &= \frac{1}{1-\varepsilon} \cdot (1 + \varepsilon - 2 \cdot \varepsilon^2 - 1) \\ &= \frac{\varepsilon}{1-\varepsilon} \cdot (1 - 2 \cdot \varepsilon) \\ &\geq 0. \end{aligned}$$