A PTAS for Minimum Makespan

Vahan Mkrtchyan[1]

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

March 24, 2014

Outline

Outline

Outline

Outline

Outline

Outline

Topics

Topics

Outline

Topics

Outline

1. Problem definition.

Topics

Outline

1. Problem definition.
2. Definition of PTAS.

Topics

Outline

1. Problem definition.
2. Definition of PTAS.
3. DP for Exact Restricted Bin Packing.

Topics

Outline

1. Problem definition.
2. Definition of PTAS.
3. DP for Exact Restricted Bin Packing.
4. The Core Algorithm.

Topics

Outline

1. Problem definition.
2. Definition of PTAS.
3. DP for Exact Restricted Bin Packing.
4. The Core Algorithm.
5. A PTAS for Minimum Makespan.

Problem definition

Problem Statement

We are given the processing times $p_1, \ldots, p_n$ of n jobs and integer m.

Problem definition

Problem Statement

We are given the processing times $p_1, \dots, p_n$ of n jobs and integer m. The goal is to find an assignment of the jobs to m identical machines, so that the final completion time (the makespan) is minimized.

Problem definition

Problem Statement

We are given the processing times $p_1, \ldots, p_n$ of n jobs and integer m. The goal is to find an assignment of the jobs to m identical machines, so that the final completion time (the makespan) is minimized.

The Bound for *OPT*

If $L = \max\{\frac{1}{m} \cdot \sum_{i=1}^{n} p_i, \max\{p_i\}\}$, then we have that: $L \leq OPT \leq 2 \cdot L$.

Problem definition

Problem Statement

We are given the processing times $p_1, \ldots, p_n$ of n jobs and integer m. The goal is to find an assignment of the jobs to m identical machines, so that the final completion time (the makespan) is minimized.

The Bound for *OPT*

If $L = \max\{\frac{1}{m} \cdot \sum_{i=1}^{n} p_i, \max\{p_i\}\}$, then we have that: $L \leq OPT \leq 2 \cdot L$.

Remark

There exists a schedule with makespan t if and only if n objects of sizes $p_1, \ldots, p_n$ can be packed into m bins of capacity t.

Problem definition

Problem Statement

We are given the processing times $p_1, \ldots, p_n$ of n jobs and integer m. The goal is to find an assignment of the jobs to m identical machines, so that the final completion time (the makespan) is minimized.

The Bound for *OPT*

If $L = \max\{\frac{1}{m} \cdot \sum_{i=1}^{n} p_i, \max\{p_i\}\}$, then we have that: $L \leq OPT \leq 2 \cdot L$.

Remark

There exists a schedule with makespan t if and only if n objects of sizes $p_1, \ldots, p_n$ can be packed into m bins of capacity t. If I is the set of sizes $p_1, \ldots, p_n$ and $bins(I, t)$ is the minimum number of bins needed to pack these objects,

Problem definition

Problem Statement

We are given the processing times $p_1, \ldots, p_n$ of n jobs and integer m. The goal is to find an assignment of the jobs to m identical machines, so that the final completion time (the makespan) is minimized.

The Bound for *OPT*

If $L = \max\{\frac{1}{m} \cdot \sum_{i=1}^{n} p_i, \max\{p_i\}\}$, then we have that: $L \leq OPT \leq 2 \cdot L$.

Remark

There exists a schedule with makespan t if and only if n objects of sizes $p_1, \ldots, p_n$ can be packed into m bins of capacity t. If I is the set of sizes $p_1, \ldots, p_n$ and $bins(I, t)$ is the minimum number of bins needed to pack these objects, then one has the following equality:

$$OPT(makespan) = \min\{t : bins(I, t) \leq m\}.$$

PTAS

Definition

A PTAS for a minimization problem Π is an algorithm A, which for 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

Definition

A PTAS for a minimization problem Π is an algorithm A, which for 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.

Restricted Instances

Some Notation

Suppose we are given n items to pack into bins of size t.

Restricted Instances

Some Notation

Suppose we are given n items to pack into bins of size t. Moreover, assume that they can have only k different sizes.

Restricted Instances

Some Notation

Suppose we are given n items to pack into bins of size t. Moreover, assume that they can have only k different sizes. Let $I = (i_1, ..., i_k)$ denote the input to such a problem, where i_j denotes the number of items j.

Restricted Instances

Some Notation

Suppose we are given n items to pack into bins of size t. Moreover, assume that they can have only k different sizes. Let $I = (i_1, \ldots, i_k)$ denote the input to such a problem, where i_j denotes the number of items j. Let $Bins(i_1, \ldots, i_k)$ be the minimum number of bins needed to pack these items.

Restricted Instances

Some Notation

Suppose we are given n items to pack into bins of size t. Moreover, assume that they can have only k different sizes. Let $I = (i_1, ..., i_k)$ denote the input to such a problem, where i_j denotes the number of items j. Let $Bins(i_1, ..., i_k)$ be the minimum number of bins needed to pack these items.

Description of DP

First compute the set Q of all k-tuples $(q_1, ..., q_k)$, such that $Bins(q_1, ..., q_k) = 1$.

Restricted Instances

Some Notation

Suppose we are given n items to pack into bins of size t. Moreover, assume that they can have only k different sizes. Let $I = (i_1, ..., i_k)$ denote the input to such a problem, where i_j denotes the number of items j. Let $Bins(i_1, ..., i_k)$ be the minimum number of bins needed to pack these items.

Description of DP

First compute the set Q of all k-tuples $(q_1, ..., q_k)$, such that $Bins(q_1, ..., q_k) = 1$. There are at most $O(n^k)$ of them and they can be found in $O(n^k)$ time.

Restricted Instances

Some Notation

Suppose we are given n items to pack into bins of size t. Moreover, assume that they can have only k different sizes. Let $I = (i_1, \ldots, i_k)$ denote the input to such a problem, where i_j denotes the number of items j. Let $Bins(i_1, \ldots, i_k)$ be the minimum number of bins needed to pack these items.

Description of DP

First compute the set Q of all k-tuples $(q_1, \ldots, q_k)$, such that $Bins(q_1, \ldots, q_k) = 1$. There are at most $O(n^k)$ of them and they can be found in $O(n^k)$ time.

- For each $q \in Q$ set $BINS(q) = 1$.

Restricted Instances

Some Notation

Suppose we are given n items to pack into bins of size t. Moreover, assume that they can have only k different sizes. Let $I = (i_1, ..., i_k)$ denote the input to such a problem, where i_j denotes the number of items j. Let $Bins(i_1, ..., i_k)$ be the minimum number of bins needed to pack these items.

Description of DP

First compute the set Q of all k-tuples $(q_1, ..., q_k)$, such that $Bins(q_1, ..., q_k) = 1$. There are at most $O(n^k)$ of them and they can be found in $O(n^k)$ time.

- For each $q \in Q$ set $BINS(q) = 1$.
- If there exists j, such that $i_j < 0$, then $Bins(i_1, ..., i_k) = +\infty$.

Restricted Instances

Some Notation

Suppose we are given n items to pack into bins of size t. Moreover, assume that they can have only k different sizes. Let $I = (i_1, \dots, i_k)$ denote the input to such a problem, where i_j denotes the number of items j. Let $Bins(i_1, \dots, i_k)$ be the minimum number of bins needed to pack these items.

Description of DP

First compute the set Q of all k-tuples $(q_1, \dots, q_k)$, such that $Bins(q_1, \dots, q_k) = 1$. There are at most $O(n^k)$ of them and they can be found in $O(n^k)$ time.

- For each $q \in Q$ set $BINS(q) = 1$.
- If there exists j, such that $i_j < 0$, then $Bins(i_1, \dots, i_k) = +\infty$.
- For all other q's use the following recurrence:

$$Bins(i_1, \dots, i_k) = 1 + \min_{(q_1, \dots, q_k) \in Q} Bins(i_1 - q_1, \dots, i_k - q_k).$$

Restricted Instances

Some Notation

Suppose we are given n items to pack into bins of size t. Moreover, assume that they can have only k different sizes. Let $I = (i_1, \ldots, i_k)$ denote the input to such a problem, where i_j denotes the number of items j. Let $Bins(i_1, \ldots, i_k)$ be the minimum number of bins needed to pack these items.

Description of DP

First compute the set Q of all k-tuples $(q_1, \ldots, q_k)$, such that $Bins(q_1, \ldots, q_k) = 1$. There are at most $O(n^k)$ of them and they can be found in $O(n^k)$ time.

- For each $q \in Q$ set $BINS(q) = 1$.
- If there exists j, such that $i_j < 0$, then $Bins(i_1, \ldots, i_k) = +\infty$.
- For all other q's use the following recurrence:

$$Bins(i_1, \ldots, i_k) = 1 + \min_{(q_1, \ldots, q_k) \in Q} Bins(i_1 - q_1, \ldots, i_k - q_k).$$

Remark

Since there are $O(n^k)$ entries, and the calculation of each entry can be carried out in time $O(n^k)$, we have that this DP runs in time $O(n^{2k})$.

Core Algorithm

Theorem

Let $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$, and $L \leq t \leq 2L$.

Core Algorithm

Theorem

Let $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$, and $L \leq t \leq 2L$. There exists an algorithm that in $O(n^{2k})$ time finds a bin packing of I that uses $\alpha(I, t, \varepsilon)$ bins of size $t \cdot (1 + \varepsilon)$.

Core Algorithm

Theorem

Let $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$, and $L \leq t \leq 2L$. There exists an algorithm that in $O(n^{2k})$ time finds a bin packing of I that uses $\alpha(I, t, \varepsilon)$ bins of size $t \cdot (1 + \varepsilon)$. The packing has the property that

$$\alpha(I, t, \varepsilon) \leq Bins(I, t)$$

for each t and ε.

Core Algorithm

Theorem

Let $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$, and $L \leq t \leq 2L$. There exists an algorithm that in $O(n^{2k})$ time finds a bin packing of I that uses $\alpha(I, t, \varepsilon)$ bins of size $t \cdot (1 + \varepsilon)$. The packing has the property that

$$\alpha(I, t, \varepsilon) \leq Bins(I, t)$$

for each t and ε.

The Core Algorithm

- An object in I is small if its size is at most $\varepsilon \cdot t$.

Core Algorithm

Theorem

Let $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$, and $L \leq t \leq 2L$. There exists an algorithm that in $O(n^{2k})$ time finds a bin packing of I that uses $\alpha(I, t, \varepsilon)$ bins of size $t \cdot (1 + \varepsilon)$. The packing has the property that

$$\alpha(I, t, \varepsilon) \leq Bins(I, t)$$

for each t and ε.

The Core Algorithm

- An object in I is small if its size is at most $\varepsilon \cdot t$.
- Round non-small objects as follows: if $p_j \in [t \cdot \varepsilon(1+\varepsilon)^i, t \cdot \varepsilon(1+\varepsilon)^{i+1}]$, then set $p'_j = t \cdot \varepsilon(1+\varepsilon)^i$.

Core Algorithm

Theorem

Let $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$, and $L \leq t \leq 2L$. There exists an algorithm that in $O(n^{2k})$ time finds a bin packing of I that uses $\alpha(I, t, \varepsilon)$ bins of size $t \cdot (1 + \varepsilon)$. The packing has the property that

$$\alpha(I, t, \varepsilon) \leq Bins(I, t)$$

for each t and ε.

The Core Algorithm

- An object in I is small if its size is at most $\varepsilon \cdot t$.
- Round non-small objects as follows: if $p_j \in [t \cdot \varepsilon(1+\varepsilon)^i, t \cdot \varepsilon(1+\varepsilon)^{i+1}]$, then set $p'_j = t \cdot \varepsilon(1+\varepsilon)^i$. There can be at most k different sizes.

Core Algorithm

Theorem

Let $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$, and $L \leq t \leq 2L$. There exists an algorithm that in $O(n^{2k})$ time finds a bin packing of I that uses $\alpha(I, t, \varepsilon)$ bins of size $t \cdot (1+\varepsilon)$. The packing has the property that

$$\alpha(I, t, \varepsilon) \leq Bins(I, t)$$

for each t and ε.

The Core Algorithm

- An object in I is small if its size is at most $\varepsilon \cdot t$.
- Round non-small objects as follows: if $p_j \in [t \cdot \varepsilon(1+\varepsilon)^i, t \cdot \varepsilon(1+\varepsilon)^{i+1}]$, then set $p'_j = t \cdot \varepsilon(1+\varepsilon)^i$. There can be at most k different sizes.
- Use the DP to pack non-small objects optimally into bins of size t using costs p'_j.

Core Algorithm

Theorem

Let $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$, and $L \leq t \leq 2L$. There exists an algorithm that in $O(n^{2k})$ time finds a bin packing of I that uses $\alpha(I, t, \varepsilon)$ bins of size $t \cdot (1+\varepsilon)$. The packing has the property that

$$\alpha(I, t, \varepsilon) \leq Bins(I, t)$$

for each t and ε.

The Core Algorithm

- An object in I is small if its size is at most $\varepsilon \cdot t$.
- Round non-small objects as follows: if $p_j \in [t \cdot \varepsilon(1+\varepsilon)^i, t \cdot \varepsilon(1+\varepsilon)^{i+1}]$, then set $p'_j = t \cdot \varepsilon(1+\varepsilon)^i$. There can be at most k different sizes.
- Use the DP to pack non-small objects optimally into bins of size t using costs p'_j. Observe that rounding can reduce the size by a factor of $1+\varepsilon$, so the resulting packing is valid for bins of size $(1+\varepsilon) \cdot t$.

Core Algorithm

Theorem

Let $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$, and $L \leq t \leq 2L$. There exists an algorithm that in $O(n^{2k})$ time finds a bin packing of I that uses $\alpha(I, t, \varepsilon)$ bins of size $t \cdot (1+\varepsilon)$. The packing has the property that

$$\alpha(I, t, \varepsilon) \leq Bins(I, t)$$

for each t and ε.

The Core Algorithm

- An object in I is small if its size is at most $\varepsilon \cdot t$.
- Round non-small objects as follows: if $p_j \in [t \cdot \varepsilon(1+\varepsilon)^i, t \cdot \varepsilon(1+\varepsilon)^{i+1}]$, then set $p'_j = t \cdot \varepsilon(1+\varepsilon)^i$. There can be at most k different sizes.
- Use the DP to pack non-small objects optimally into bins of size t using costs p'_j. Observe that rounding can reduce the size by a factor of $1+\varepsilon$, so the resulting packing is valid for bins of size $(1+\varepsilon) \cdot t$.
- Apply First -Fit to the resulting packing for small items.

Core Algorithm: Proof of Correctness

The Proof

If the algorithm opens new bins, then all of the bins except possibly the last one are filled to at least size t.

Core Algorithm: Proof of Correctness

The Proof

If the algorithm opens new bins, then all of the bins except possibly the last one are filled to at least size t. Thus the optimal packing into bins of size t must use at least $\alpha(I, t, \varepsilon)$ bins.

Core Algorithm: Proof of Correctness

The Proof

If the algorithm opens new bins, then all of the bins except possibly the last one are filled to at least size t. Thus the optimal packing into bins of size t must use at least $\alpha(I,t,\varepsilon)$ bins. On the other hand, if the algorithm does not open new bins, then let I' be the set of non-small items.

Core Algorithm: Proof of Correctness

The Proof

If the algorithm opens new bins, then all of the bins except possibly the last one are filled to at least size t. Thus the optimal packing into bins of size t must use at least $\alpha(I,t,\varepsilon)$ bins. On the other hand, if the algorithm does not open new bins, then let I' be the set of non-small items. Then:

$$\alpha(I,t,\varepsilon) \quad =$$

Core Algorithm: Proof of Correctness

The Proof

If the algorithm opens new bins, then all of the bins except possibly the last one are filled to at least size t. Thus the optimal packing into bins of size t must use at least $\alpha(I,t,\varepsilon)$ bins. On the other hand, if the algorithm does not open new bins, then let I' be the set of non-small items. Then:

$$\alpha(I,t,\varepsilon) \quad = \quad \alpha(I',t,\varepsilon)$$

Core Algorithm: Proof of Correctness

The Proof

If the algorithm opens new bins, then all of the bins except possibly the last one are filled to at least size t. Thus the optimal packing into bins of size t must use at least $\alpha(I,t,\varepsilon)$ bins. On the other hand, if the algorithm does not open new bins, then let I' be the set of non-small items. Then:

$$\begin{aligned} \alpha(I,t,\varepsilon) &= \alpha(I',t,\varepsilon) \\ &\leq \end{aligned}$$

Core Algorithm: Proof of Correctness

The Proof

If the algorithm opens new bins, then all of the bins except possibly the last one are filled to at least size t. Thus the optimal packing into bins of size t must use at least $\alpha(I, t, \varepsilon)$ bins. On the other hand, if the algorithm does not open new bins, then let I' be the set of non-small items. Then:

$$\begin{aligned} \alpha(I, t, \varepsilon) &= \alpha(I', t, \varepsilon) \\ &\leq bins(I', t) \end{aligned}$$

Core Algorithm: Proof of Correctness

The Proof

If the algorithm opens new bins, then all of the bins except possibly the last one are filled to at least size t. Thus the optimal packing into bins of size t must use at least $\alpha(I, t, \varepsilon)$ bins. On the other hand, if the algorithm does not open new bins, then let I' be the set of non-small items. Then:

$$\begin{aligned} \alpha(I, t, \varepsilon) &= \alpha(I', t, \varepsilon) \\ &\leq bins(I', t) \\ &\leq \end{aligned}$$

Core Algorithm: Proof of Correctness

The Proof

If the algorithm opens new bins, then all of the bins except possibly the last one are filled to at least size t. Thus the optimal packing into bins of size t must use at least $\alpha(I,t,\varepsilon)$ bins. On the other hand, if the algorithm does not open new bins, then let I' be the set of non-small items. Then:

$$\begin{aligned}\alpha(I,t,\varepsilon) &= \alpha(I',t,\varepsilon)\\ &\leq bins(I',t)\\ &\leq bins(I,t).\end{aligned}$$

The General Case

Theorem

There exists an algorithm A, such that for each $\varepsilon > 0$ it finds a schedule with makespan at most $(1 + 3 \cdot \varepsilon) \cdot OPT$.

The General Case

Theorem

There exists an algorithm A, such that for each $\varepsilon > 0$ it finds a schedule with makespan at most $(1 + 3 \cdot \varepsilon) \cdot OPT$. The running time of the algorithm is $O(n^{2k} \cdot \lceil \log_2 \frac{1}{\varepsilon} \rceil)$, where $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$.

The General Case

Theorem

There exists an algorithm A, such that for each $\varepsilon > 0$ it finds a schedule with makespan at most $(1 + 3 \cdot \varepsilon) \cdot OPT$. The running time of the algorithm is $O(n^{2k} \cdot \lceil \log_2 \frac{1}{\varepsilon} \rceil)$, where $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$. In other words, bin packing problem admits a PTAS.

The General Case

Theorem

There exists an algorithm A, such that for each $\varepsilon > 0$ it finds a schedule with makespan at most $(1 + 3 \cdot \varepsilon) \cdot OPT$. The running time of the algorithm is $O(n^{2k} \cdot \lceil \log_2 \frac{1}{\varepsilon} \rceil)$, where $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$. In other words, bin packing problem admits a PTAS.

The Algorithm

- If $\alpha(I, L, \varepsilon) \leq m$, then use packing given by core algorithm for $t = L$.

The General Case

Theorem

There exists an algorithm A, such that for each $\varepsilon > 0$ it finds a schedule with makespan at most $(1 + 3 \cdot \varepsilon) \cdot OPT$. The running time of the algorithm is $O(n^{2k} \cdot \lceil \log_2 \frac{1}{\varepsilon} \rceil)$, where $k = \lceil \log_{1+\varepsilon} \frac{1}{\varepsilon} \rceil$. In other words, bin packing problem admits a PTAS.

The Algorithm

- If $\alpha(I, L, \varepsilon) \leq m$, then use packing given by core algorithm for $t = L$.
- If $\alpha(I, L, \varepsilon) > m$, then perform a binary search to find an interval $[T', T] \subseteq [L, 2 \cdot L]$ with $T - T' \leq \varepsilon \cdot L$, such that $\alpha(I, T', \varepsilon) > m$ and $\alpha(I, T, \varepsilon) \leq m$.

The General Case

Theorem

There exists an algorithm A, such that for each $\varepsilon > 0$ it finds a schedule with makespan at most $(1+3\cdot\varepsilon)\cdot OPT$. The running time of the algorithm is $O(n^{2k}\cdot\lceil\log_2\frac{1}{\varepsilon}\rceil)$, where $k = \lceil\log_{1+\varepsilon}\frac{1}{\varepsilon}\rceil$. In other words, bin packing problem admits a PTAS.

The Algorithm

- If $\alpha(I,L,\varepsilon) \leq m$, then use packing given by core algorithm for $t = L$.
- If $\alpha(I,L,\varepsilon) > m$, then perform a binary search to find an interval $[T', T] \subseteq [L, 2\cdot L]$ with $T - T' \leq \varepsilon\cdot L$, such that $\alpha(I,T',\varepsilon) > m$ and $\alpha(I,T,\varepsilon) \leq m$. Then return the packing given by core algorithm for $t = T$.

The Analysis of the Algorithm

The Analysis: Running Time

The binary search uses at most $\lceil \log_2 \frac{1}{\varepsilon} \rceil$ steps, hence the running time is as in the statement of the theorem.

The Analysis of the Algorithm

The Analysis: Running Time

The binary search uses at most $\lceil \log_2 \frac{1}{\varepsilon} \rceil$ steps, hence the running time is as in the statement of the theorem.

The Analysis: Case $\alpha(I, L, \varepsilon) \leq m$

Then the makespan returned by the algorithm is at most

$$\leq$$

The Analysis of the Algorithm

The Analysis: Running Time

The binary search uses at most $\lceil \log_2 \frac{1}{\varepsilon} \rceil$ steps, hence the running time is as in the statement of the theorem.

The Analysis: Case $\alpha(I, L, \varepsilon) \leq m$

Then the makespan returned by the algorithm is at most

$$\leq \quad L \cdot (1 + \varepsilon)$$

The Analysis of the Algorithm

The Analysis: Running Time

The binary search uses at most $\lceil \log_2 \frac{1}{\varepsilon} \rceil$ steps, hence the running time is as in the statement of the theorem.

The Analysis: Case $\alpha(I, L, \varepsilon) \leq m$

Then the makespan returned by the algorithm is at most

$$\begin{aligned} &\leq L \cdot (1 + \varepsilon) \\ &\leq \end{aligned}$$

The Analysis of the Algorithm

The Analysis: Running Time

The binary search uses at most $\lceil \log_2 \frac{1}{\varepsilon} \rceil$ steps, hence the running time is as in the statement of the theorem.

The Analysis: Case $\alpha(I, L, \varepsilon) \leq m$

Then the makespan returned by the algorithm is at most

$$\begin{aligned} &\leq L \cdot (1 + \varepsilon) \\ &\leq (1 + \varepsilon) \cdot OPT \end{aligned}$$

The Analysis of the Algorithm

The Analysis: Running Time

The binary search uses at most $\lceil \log_2 \frac{1}{\varepsilon} \rceil$ steps, hence the running time is as in the statement of the theorem.

The Analysis: Case $\alpha(I, L, \varepsilon) \leq m$

Then the makespan returned by the algorithm is at most

$$\begin{aligned} &\leq L \cdot (1+\varepsilon) \\ &\leq (1+\varepsilon) \cdot OPT \\ &\leq \end{aligned}$$

The Analysis of the Algorithm

The Analysis: Running Time

The binary search uses at most $\lceil \log_2 \frac{1}{\varepsilon} \rceil$ steps, hence the running time is as in the statement of the theorem.

The Analysis: Case $\alpha(I, L, \varepsilon) \leq m$

Then the makespan returned by the algorithm is at most

$$\begin{aligned} &\leq L \cdot (1 + \varepsilon) \\ &\leq (1 + \varepsilon) \cdot OPT \\ &\leq (1 + 3 \cdot \varepsilon) \cdot OPT. \end{aligned}$$

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$ and

$$T \leq$$

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$ and

$$T \leq T' + \varepsilon \cdot L$$

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$ and

$$\begin{aligned} T &\leq T' + \varepsilon \cdot L \\ &\leq \end{aligned}$$

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$ and

$$\begin{aligned} T &\leq T' + \varepsilon \cdot L \\ &\leq OPT + \varepsilon \cdot OPT \end{aligned}$$

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$ and

$$\begin{aligned} T &\leq T' + \varepsilon \cdot L \\ &\leq OPT + \varepsilon \cdot OPT \\ &= \end{aligned}$$

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$ and

$$\begin{aligned} T &\leq T' + \varepsilon \cdot L \\ &\leq OPT + \varepsilon \cdot OPT \\ &= (1+\varepsilon) \cdot OPT. \end{aligned}$$

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$ and

$$\begin{aligned} T &\leq T' + \varepsilon \cdot L \\ &\leq OPT + \varepsilon \cdot OPT \\ &= (1 + \varepsilon) \cdot OPT. \end{aligned}$$

Since the core algorithm for $T = t$ returns a schedule with makespan at most $(1 + \varepsilon) \cdot T$,

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$ and

$$\begin{aligned} T &\leq T' + \varepsilon \cdot L \\ &\leq OPT + \varepsilon \cdot OPT \\ &= (1+\varepsilon) \cdot OPT. \end{aligned}$$

Since the core algorithm for $T = t$ returns a schedule with makespan at most $(1+\varepsilon) \cdot T$, the makespan of the returned schedule is at most

$$(1+\varepsilon) \cdot T \quad \leq$$

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$ and

$$\begin{aligned} T &\leq T' + \varepsilon \cdot L \\ &\leq OPT + \varepsilon \cdot OPT \\ &= (1+\varepsilon) \cdot OPT. \end{aligned}$$

Since the core algorithm for $T = t$ returns a schedule with makespan at most $(1+\varepsilon) \cdot T$, the makespan of the returned schedule is at most

$$(1+\varepsilon) \cdot T \leq (1+\varepsilon)^2 \cdot OPT$$

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$ and

$$\begin{aligned} T &\leq T' + \varepsilon \cdot L \\ &\leq OPT + \varepsilon \cdot OPT \\ &= (1 + \varepsilon) \cdot OPT. \end{aligned}$$

Since the core algorithm for $T = t$ returns a schedule with makespan at most $(1 + \varepsilon) \cdot T$, the makespan of the returned schedule is at most

$$\begin{aligned} (1 + \varepsilon) \cdot T &\leq (1 + \varepsilon)^2 \cdot OPT \\ &\leq \end{aligned}$$

The Analysis of the Algorithm

The Analysis: Case $\alpha(I, L, \varepsilon) > m$

$m < \alpha(I, T', \varepsilon) \leq bins(I, T')$, so $T' \leq OPT$ and

$$\begin{aligned} T &\leq T' + \varepsilon \cdot L \\ &\leq OPT + \varepsilon \cdot OPT \\ &= (1 + \varepsilon) \cdot OPT. \end{aligned}$$

Since the core algorithm for $T = t$ returns a schedule with makespan at most $(1 + \varepsilon) \cdot T$, the makespan of the returned schedule is at most

$$\begin{aligned} (1 + \varepsilon) \cdot T &\leq (1 + \varepsilon)^2 \cdot OPT \\ &\leq (1 + 3 \cdot \varepsilon) \cdot OPT. \end{aligned}$$