Minimizing Makespan on identical parallel machines

K. Subramani[1]

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

March 18, 2014

Outline

Outline

Outline

Topics

Topics

Outline

Topics

Outline

1. Problem definition.

Topics

Outline

1. Problem definition.
2. The 2 approximation algorithm.

Topics

Outline

1. Problem definition.
2. The 2 approximation algorithm.
3. The LPT rule.

Problem definition

Problem Statement

Problem definition

Problem Statement

1. We are given m identical machines $\{M_1, M_2, \ldots M_m\}$ and a collection of n jobs, $\{J_1, J_2, \ldots, J_n\}$, with respective processing times $\{p_1, p_2, \ldots p_n\}$.

Problem definition

Problem Statement

1. We are given m identical machines $\{M_1, M_2, \ldots M_m\}$ and a collection of n jobs, $\{J_1, J_2, \ldots, J_n\}$, with respective processing times $\{p_1, p_2, \ldots p_n\}$.
2. A schedule is an assignment of jobs to machines.

Problem definition

Problem Statement

1. We are given m identical machines $\{M_1, M_2, \ldots M_m\}$ and a collection of n jobs, $\{J_1, J_2, \ldots, J_n\}$, with respective processing times $\{p_1, p_2, \ldots p_n\}$.
2. A schedule is an assignment of jobs to machines.
3. The makespan of a schedule is the time at which the last job completes.

Problem definition

Problem Statement

1. We are given m identical machines $\{M_1, M_2, \ldots M_m\}$ and a collection of n jobs, $\{J_1, J_2, \ldots, J_n\}$, with respective processing times $\{p_1, p_2, \ldots p_n\}$.
2. A schedule is an assignment of jobs to machines.
3. The makespan of a schedule is the time at which the last job completes.
4. The jobs are non-peemptive and the order of jobs on a machine does not affect the makespan.

Problem definition

Problem Statement

1. We are given m identical machines $\{M_1, M_2, \ldots M_m\}$ and a collection of n jobs, $\{J_1, J_2, \ldots, J_n\}$, with respective processing times $\{p_1, p_2, \ldots p_n\}$.
2. A schedule is an assignment of jobs to machines.
3. The makespan of a schedule is the time at which the last job completes.
4. The jobs are non-peemptive and the order of jobs on a machine does not affect the makespan.
5. Depending upon the assignment of jobs to machines, the same input may not have a unique makespan.

Problem definition

Problem Statement

1. We are given m identical machines $\{M_1, M_2, \ldots M_m\}$ and a collection of n jobs, $\{J_1, J_2, \ldots, J_n\}$, with respective processing times $\{p_1, p_2, \ldots p_n\}$.
2. A schedule is an assignment of jobs to machines.
3. The makespan of a schedule is the time at which the last job completes.
4. The jobs are non-peemptive and the order of jobs on a machine does not affect the makespan.
5. Depending upon the assignment of jobs to machines, the same input may not have a unique makespan.
6. The goal is to find the assignment in which the makespan is minimized.

Problem definition

Problem Statement

1. We are given m identical machines $\{M_1, M_2, \ldots M_m\}$ and a collection of n jobs, $\{J_1, J_2, \ldots, J_n\}$, with respective processing times $\{p_1, p_2, \ldots p_n\}$.
2. A schedule is an assignment of jobs to machines.
3. The makespan of a schedule is the time at which the last job completes.
4. The jobs are non-peemptive and the order of jobs on a machine does not affect the makespan.
5. Depending upon the assignment of jobs to machines, the same input may not have a unique makespan.
6. The goal is to find the assignment in which the makespan is minimized.

Note

Can you solve the problem when $m = 1$?

Problem definition

Problem Statement

1. We are given m identical machines $\{M_1, M_2, \ldots M_m\}$ and a collection of n jobs, $\{J_1, J_2, \ldots, J_n\}$, with respective processing times $\{p_1, p_2, \ldots p_n\}$.
2. A schedule is an assignment of jobs to machines.
3. The makespan of a schedule is the time at which the last job completes.
4. The jobs are non-peemptive and the order of jobs on a machine does not affect the makespan.
5. Depending upon the assignment of jobs to machines, the same input may not have a unique makespan.
6. The goal is to find the assignment in which the makespan is minimized.

Note

Can you solve the problem when $m = 1$? How about when $m = 2$?

The List Scheduling Algorithm

List Scheduling

The List Scheduling Algorithm

List Scheduling

1. **for** ($i = 1$ **to** n)

The List Scheduling Algorithm

List Scheduling

1. **for** ($i = 1$ **to** n)
2. Assign job J_i to the least loaded machine.

The List Scheduling Algorithm

List Scheduling

1. **for** ($i = 1$ **to** n)
2. Assign job J_i to the least loaded machine.
3. **endfor**

Analysis of List Scheduling

Analysis of List Scheduling

Analysis

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.

Analysis of List Scheduling

Analysis

1. Let *OPT* denote the makespan of the optimal schedule and let *LS* denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last.

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.
3. Consider the m machines, before J_l was assigned.

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.
3. Consider the m machines, before J_l was assigned. Clearly, all machines much have jobs that finish at or after t_l.

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.
3. Consider the m machines, before J_l was assigned. Clearly, all machines much have jobs that finish at or after t_l.
4. Hence, $t_l \leq \frac{1}{m} \sum_{i=1, i \neq l}^{n} p_i$.

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.
3. Consider the m machines, before J_l was assigned. Clearly, all machines much have jobs that finish at or after t_l.
4. Hence, $t_l \leq \frac{1}{m} \sum_{i=1, i \neq l}^{n} p_i$. Likewise, we must have $OPT \geq \frac{1}{m} \sum_{i=1}^{n} p_i$.

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.
3. Consider the m machines, before J_l was assigned. Clearly, all machines much have jobs that finish at or after t_l.
4. Hence, $t_l \leq \frac{1}{m}\sum_{i=1, i\neq l}^{n} p_i$. Likewise, we must have $OPT \geq \frac{1}{m}\sum_{i=1}^{n} p_i$.
5.

$$LS \quad =$$

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.
3. Consider the m machines, before J_l was assigned. Clearly, all machines much have jobs that finish at or after t_l.
4. Hence, $t_l \leq \frac{1}{m}\sum_{i=1,i\neq l}^{n} p_i$. Likewise, we must have $OPT \geq \frac{1}{m}\sum_{i=1}^{n} p_i$.
5.

$$LS = t_l + p_l$$

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.
3. Consider the m machines, before J_l was assigned. Clearly, all machines much have jobs that finish at or after t_l.
4. Hence, $t_l \leq \frac{1}{m}\sum_{i=1,i\neq l}^{n} p_i$. Likewise, we must have $OPT \geq \frac{1}{m}\sum_{i=1}^{n} p_i$.
5.

$$\begin{aligned} LS &= t_l + p_l \\ &\leq \end{aligned}$$

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.
3. Consider the m machines, before J_l was assigned. Clearly, all machines much have jobs that finish at or after t_l.
4. Hence, $t_l \leq \frac{1}{m}\sum_{i=1,i\neq l}^{n} p_i$. Likewise, we must have $OPT \geq \frac{1}{m}\sum_{i=1}^{n} p_i$.
5.

$$\begin{aligned} LS &= t_l + p_l \\ &\leq \frac{1}{m}\sum_{i=1,i\neq l}^{n} p_i + p_l \end{aligned}$$

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.
3. Consider the m machines, before J_l was assigned. Clearly, all machines much have jobs that finish at or after t_l.
4. Hence, $t_l \leq \frac{1}{m}\sum_{i=1,i\neq l}^{n} p_i$. Likewise, we must have $OPT \geq \frac{1}{m}\sum_{i=1}^{n} p_i$.
5.

$$\begin{aligned} LS &= t_l + p_l \\ &\leq \frac{1}{m}\sum_{i=1,i\neq l}^{n} p_i + p_l \\ &= \frac{1}{m}\sum_{i=1}^{n} p_l + p_i(1-\frac{1}{m}) \end{aligned}$$

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.
3. Consider the m machines, before J_l was assigned. Clearly, all machines much have jobs that finish at or after t_l.
4. Hence, $t_l \leq \frac{1}{m}\sum_{i=1,i\neq l}^{n} p_i$. Likewise, we must have $OPT \geq \frac{1}{m}\sum_{i=1}^{n} p_i$.
5.

$$\begin{aligned} LS &= t_l + p_l \\ &\leq \frac{1}{m}\sum_{i=1,i\neq l}^{n} p_i + p_l \\ &= \frac{1}{m}\sum_{i=1}^{n} p_l + p_i(1-\frac{1}{m}) \\ &\leq OPT + OPT\cdot(1-\frac{1}{m}) = \end{aligned}$$

Analysis of List Scheduling

Analysis

1. Let OPT denote the makespan of the optimal schedule and let LS denote the makespan of the list schedule.
2. Let J_l denote the job that determines the makespan in the List Scheduling algorithm, i.e., J_l is the job that finishes last. Let t_l denote the start time of J_l.
3. Consider the m machines, before J_l was assigned. Clearly, all machines much have jobs that finish at or after t_l.
4. Hence, $t_l \leq \frac{1}{m}\sum_{i=1,i\neq l}^{n} p_i$. Likewise, we must have $OPT \geq \frac{1}{m}\sum_{i=1}^{n} p_i$.
5.

$$\begin{aligned} LS &= t_l + p_l \\ &\leq \frac{1}{m}\sum_{i=1,i\neq l}^{n} p_i + p_l \\ &= \frac{1}{m}\sum_{i=1}^{n} p_l + p_i(1-\frac{1}{m}) \\ &\leq OPT + OPT\cdot(1-\frac{1}{m}) = (2-\frac{1}{m})\cdot OPT \end{aligned}$$

Tightness Analysis

Tight Example

Tightness Analysis

Tight Example

Consider m machines, $m \cdot (m-1)$ jobs of size 1 and one large job of size m.

The Longest Processing Time first rule (LPT)

The Longest Processing Time first rule (LPT)

LPT

The Longest Processing Time first rule (LPT)

LPT

1. Sort the jobs by processing time in descending order, so that $p_n \geq p_{n-1} \geq \ldots p_1$.

The Longest Processing Time first rule (LPT)

LPT

1. Sort the jobs by processing time in descending order, so that $p_n \geq p_{n-1} \geq \ldots p_1$.
2. **for** ($i = n$ **downto** 1)

The Longest Processing Time first rule (LPT)

LPT

1. Sort the jobs by processing time in descending order, so that $p_n \geq p_{n-1} \geq \ldots p_1$.
2. **for** ($i = n$ **downto** 1)
3. Assign job J_i to the least loaded machine.

The Longest Processing Time first rule (LPT)

LPT

1. Sort the jobs by processing time in descending order, so that $p_n \geq p_{n-1} \geq \ldots p_1$.
2. **for** ($i = n$ **downto** 1)
3. Assign job J_i to the least loaded machine.
4. **endfor**

The Longest Processing Time first rule (LPT)

LPT

1. Sort the jobs by processing time in descending order, so that $p_n \geq p_{n-1} \geq \ldots p_1$.
2. **for** ($i = n$ **downto** 1)
3. Assign job J_i to the least loaded machine.
4. **endfor**

Note

The Longest Processing Time first rule (LPT)

LPT

1. Sort the jobs by processing time in descending order, so that $p_n \geq p_{n-1} \geq \ldots p_1$.
2. **for** ($i = n$ **downto** 1)
3. Assign job J_i to the least loaded machine.
4. **endfor**

Note

LPT is not optimal.

The Longest Processing Time first rule (LPT)

LPT

1. Sort the jobs by processing time in descending order, so that $p_n \geq p_{n-1} \geq \ldots p_1$.
2. **for** ($i = n$ **downto** 1)
3. Assign job J_i to the least loaded machine.
4. **endfor**

Note

LPT is not optimal. $m = 3$ *and* $\mathbf{p} = (8,5,4,3,3,3)$.

Analysis of LPT

Analysis of LPT

Note

Let LS_{LPT} denote the makespan returned by LPT and let OPT denote the optimal solution.

Analysis of LPT

Note

Let LS_{LPT} denote the makespan returned by LPT and let OPT denote the optimal solution.

Observation

If $n \leq m$, LPT is optimal.

Analysis of LPT

Note

Let LS_{LPT} denote the makespan returned by LPT and let OPT denote the optimal solution.

Observation

If $n \leq m$, LPT is optimal.

Observation

If $n > m$, then $OPT \geq 2 \cdot p_{m+1}$.

Analysis of LPT

Note

Let LS_{LPT} denote the makespan returned by LPT and let OPT denote the optimal solution.

Observation

If $n \leq m$, LPT is optimal.

Observation

If $n > m$, then $OPT \geq 2 \cdot p_{m+1}$.

Proof.

Pigeonhole principle. □

Analysis of LPT

Note

Let LS_{LPT} denote the makespan returned by LPT and let OPT denote the optimal solution.

Observation

If $n \leq m$, LPT is optimal.

Observation

If $n > m$, then $OPT \geq 2 \cdot p_{m+1}$.

Proof.

Pigeonhole principle. □

Theorem

LPT is a $\frac{3}{2}$ approximation algorithm.

Proof of $\frac{3}{2}$ approximation

Proof of $\frac{3}{2}$ approximation

Proof.

The makespan is determined by one of the jobs in $\{J_{m+1}, \ldots, J_n\}$.

Proof of $\frac{3}{2}$ approximation

Proof.

The makespan is determined by one of the jobs in $\{J_{m+1}, \ldots, J_n\}$. Let J_l denote the job finishing last and let t_l denote its start time.

Proof of $\frac{3}{2}$ approximation

Proof.

The makespan is determined by one of the jobs in $\{J_{m+1}, \ldots, J_n\}$. Let J_l denote the job finishing last and let t_l denote its start time. We know that

Proof of $\frac{3}{2}$ approximation

Proof.

The makespan is determined by one of the jobs in $\{J_{m+1}, \ldots, J_n\}$. Let J_l denote the job finishing last and let t_l denote its start time. We know that

$$LS_{LPT} \quad =$$

Proof of $\frac{3}{2}$ approximation

Proof.

The makespan is determined by one of the jobs in $\{J_{m+1}, \ldots, J_n\}$. Let J_l denote the job finishing last and let t_l denote its start time. We know that

$$LS_{LPT} = t_l + p_l$$

Proof of $\frac{3}{2}$ approximation

Proof.

The makespan is determined by one of the jobs in $\{J_{m+1}, \ldots, J_n\}$. Let J_l denote the job finishing last and let t_l denote its start time. We know that

$$\begin{aligned} LS_{LPT} &= t_l + p_l \\ &\leq \end{aligned}$$

Proof of $\frac{3}{2}$ approximation

Proof.

The makespan is determined by one of the jobs in $\{J_{m+1}, \ldots, J_n\}$. Let J_l denote the job finishing last and let t_l denote its start time. We know that

$$\begin{aligned} LS_{LPT} &= t_l + p_l \\ &\leq \frac{1}{m} \sum_{i=1, i \neq l}^{n} p_i + p_l \end{aligned}$$

Proof of $\frac{3}{2}$ approximation

Proof.

The makespan is determined by one of the jobs in $\{J_{m+1}, \ldots, J_n\}$. Let J_l denote the job finishing last and let t_l denote its start time. We know that

$$\begin{aligned} LS_{LPT} &= t_l + p_l \\ &\leq \frac{1}{m} \sum_{i=1, i \neq l}^{n} p_i + p_l \\ &= \frac{1}{m} \sum_{i=1}^{n} p_l + p_i (1 - \frac{1}{m}) \end{aligned}$$

Proof of $\frac{3}{2}$ approximation

Proof.

The makespan is determined by one of the jobs in $\{J_{m+1}, \ldots, J_n\}$. Let J_l denote the job finishing last and let t_l denote its start time. We know that

$$\begin{aligned} LS_{LPT} &= t_l + p_l \\ &\leq \frac{1}{m} \sum_{i=1, i \neq l}^{n} p_i + p_l \\ &= \frac{1}{m} \sum_{i=1}^{n} p_l + p_i (1 - \frac{1}{m}) \\ &\leq OPT + \frac{OPT}{2} \cdot (1 - \frac{1}{m}) \end{aligned}$$

Proof of $\frac{3}{2}$ approximation

Proof.

The makespan is determined by one of the jobs in $\{J_{m+1}, \ldots, J_n\}$. Let J_l denote the job finishing last and let t_l denote its start time. We know that

$$\begin{aligned} LS_{LPT} &= t_l + p_l \\ &\leq \frac{1}{m} \sum_{i=1, i \neq l}^{n} p_i + p_l \\ &= \frac{1}{m} \sum_{i=1}^{n} p_l + p_i (1 - \frac{1}{m}) \\ &\leq OPT + \frac{OPT}{2} \cdot (1 - \frac{1}{m}) \\ &\leq \frac{3}{2} \cdot OPT \end{aligned}$$

□

Tighter Analysis

Analysis of LPT

Tighter Analysis

Analysis of LPT

1. Let LS_{LPT} denote the makespan returned by LPT.

Tighter Analysis

Analysis of LPT

1. Let LS_{LPT} denote the makespan returned by LPT.
2. Let J_l with processing time p_l be the job that finishes last.

Tighter Analysis

Analysis of LPT

1. Let LS_{LPT} denote the makespan returned by LPT.
2. Let J_l with processing time p_l be the job that finishes last. We can actually assume that J_l is the last job.

Tighter Analysis

Analysis of LPT

1. Let LS_{LPT} denote the makespan returned by LPT.
2. Let J_l with processing time p_l be the job that finishes last. We can actually assume that J_l is the last job.
3. Recall that for any list scheduling algorithm (including LPT), we must have,

Tighter Analysis

Analysis of LPT

1. Let LS_{LPT} denote the makespan returned by LPT.
2. Let J_l with processing time p_l be the job that finishes last. We can actually assume that J_l is the last job.
3. Recall that for any list scheduling algorithm (including LPT), we must have,

$$LS_{LPT} \leq \frac{1}{m} \cdot \sum_{i=1}^{n} p_i + p_l \cdot (1 - \frac{1}{m})$$

Tighter Analysis

Analysis of LPT

1. Let LS_{LPT} denote the makespan returned by LPT.
2. Let J_l with processing time p_l be the job that finishes last. We can actually assume that J_l is the last job.
3. Recall that for any list scheduling algorithm (including LPT), we must have,

$$\begin{aligned} LS_{LPT} &\leq \frac{1}{m} \cdot \sum_{i=1}^{n} p_i + p_l \cdot (1 - \frac{1}{m}) \\ &= OPT + p_l \cdot (1 - \frac{1}{m}) \end{aligned}$$

Tighter Analysis

Analysis of LPT

1. Let LS_{LPT} denote the makespan returned by LPT.
2. Let J_l with processing time p_l be the job that finishes last. We can actually assume that J_l is the last job.
3. Recall that for any list scheduling algorithm (including LPT), we must have,

$$\begin{aligned} LS_{LPT} &\leq \frac{1}{m} \cdot \sum_{i=1}^{n} p_i + p_l \cdot (1 - \frac{1}{m}) \\ &= OPT + p_l \cdot (1 - \frac{1}{m}) \end{aligned}$$

4. What happens if $p_l \leq \frac{OPT}{3}$?

Tighter Analysis

Analysis of LPT

1. Let LS_{LPT} denote the makespan returned by LPT.
2. Let J_l with processing time p_l be the job that finishes last. We can actually assume that J_l is the last job.
3. Recall that for any list scheduling algorithm (including LPT), we must have,

$$\begin{aligned} LS_{LPT} &\leq \frac{1}{m} \cdot \sum_{i=1}^{n} p_i + p_l \cdot (1 - \frac{1}{m}) \\ &= OPT + p_l \cdot (1 - \frac{1}{m}) \end{aligned}$$

4. What happens if $p_l \leq \frac{OPT}{3}$? In this case,

$$LS_{LPT} \leq OPT + \frac{OPT}{3} \cdot (1 - \frac{1}{m})$$

Tighter Analysis

Analysis of LPT

1. Let LS_{LPT} denote the makespan returned by LPT.
2. Let J_l with processing time p_l be the job that finishes last. We can actually assume that J_l is the last job.
3. Recall that for any list scheduling algorithm (including LPT), we must have,

$$\begin{aligned} LS_{LPT} &\leq \frac{1}{m} \cdot \sum_{i=1}^{n} p_i + p_l \cdot (1 - \frac{1}{m}) \\ &= OPT + p_l \cdot (1 - \frac{1}{m}) \end{aligned}$$

4. What happens if $p_l \leq \frac{OPT}{3}$? In this case,

$$\begin{aligned} LS_{LPT} &\leq OPT + \frac{OPT}{3} \cdot (1 - \frac{1}{m}) \\ &\leq OPT + \frac{OPT}{3} \end{aligned}$$

Tighter Analysis

Analysis of LPT

1. Let LS_{LPT} denote the makespan returned by LPT.
2. Let J_l with processing time p_l be the job that finishes last. We can actually assume that J_l is the last job.
3. Recall that for any list scheduling algorithm (including LPT), we must have,

$$\begin{aligned} LS_{LPT} &\leq \frac{1}{m} \cdot \sum_{i=1}^{n} p_i + p_l \cdot (1 - \frac{1}{m}) \\ &= OPT + p_l \cdot (1 - \frac{1}{m}) \end{aligned}$$

4. What happens if $p_l \leq \frac{OPT}{3}$? In this case,

$$\begin{aligned} LS_{LPT} &\leq OPT + \frac{OPT}{3} \cdot (1 - \frac{1}{m}) \\ &\leq OPT + \frac{OPT}{3} \\ &= \frac{4}{3} \cdot OPT \end{aligned}$$

Tighter Analysis of LPT

Tighter Analysis of LPT

LPT

Tighter Analysis of LPT

LPT

What happens if $p_l > \frac{OPT}{3}$?

Tighter Analysis of LPT

LPT

What happens if $p_l > \frac{OPT}{3}$? In this case, LPT is optimal, i.e., $LS_{LPT} = OPT$!

Tighter Analysis of LPT

LPT

What happens if $p_l > \frac{OPT}{3}$? In this case, LPT is optimal, i.e., $LS_{LPT} = OPT$!

1. Focus on the optimum schedule.

Tighter Analysis of LPT

LPT

What happens if $p_l > \frac{OPT}{3}$? In this case, LPT is optimal, i.e., $LS_{LPT} = OPT$!

1. Focus on the optimum schedule.
2. Each machine must have at least one and at most two jobs.

Tighter Analysis of LPT

LPT

What happens if $p_l > \frac{OPT}{3}$? In this case, LPT is optimal, i.e., $LS_{LPT} = OPT$!

1. Focus on the optimum schedule.
2. Each machine must have at least one and at most two jobs. (Why?)

Tighter Analysis of LPT

LPT

What happens if $p_l > \frac{OPT}{3}$? In this case, LPT is optimal, i.e., $LS_{LPT} = OPT$!

1. Focus on the optimum schedule.
2. Each machine must have at least one and at most two jobs. (Why?) Recall that J_l is the last job in the sequence.

Tighter Analysis of LPT

LPT

What happens if $p_l > \frac{OPT}{3}$? In this case, LPT is optimal, i.e., $LS_{LPT} = OPT$!

1. Focus on the optimum schedule.
2. Each machine must have at least one and at most two jobs. (Why?) Recall that J_l is the last job in the sequence.
3. W.l.o.g. assume that there are exactly $2 \cdot m$ jobs.

Tighter Analysis of LPT

LPT

What happens if $p_l > \frac{OPT}{3}$? In this case, LPT is optimal, i.e., $LS_{LPT} = OPT$!

1. Focus on the optimum schedule.
2. Each machine must have at least one and at most two jobs. (Why?) Recall that J_l is the last job in the sequence.
3. W.l.o.g. assume that there are exactly $2 \cdot m$ jobs.
4. Focus on the structure of the jobs in the optimal solution.

Tighter Analysis of LPT

LPT

What happens if $p_l > \frac{OPT}{3}$? In this case, LPT is optimal, i.e., $LS_{LPT} = OPT$!

1. Focus on the optimum schedule.
2. Each machine must have at least one and at most two jobs. (Why?) Recall that J_l is the last job in the sequence.
3. W.l.o.g. assume that there are exactly $2 \cdot m$ jobs.
4. Focus on the structure of the jobs in the optimal solution. Is it different from LPT?

Tighter Analysis of LPT

LPT

What happens if $p_l > \frac{OPT}{3}$? In this case, LPT is optimal, i.e., $LS_{LPT} = OPT$!

1. Focus on the optimum schedule.
2. Each machine must have at least one and at most two jobs. (Why?) Recall that J_l is the last job in the sequence.
3. W.l.o.g. assume that there are exactly $2 \cdot m$ jobs.
4. Focus on the structure of the jobs in the optimal solution. Is it different from LPT? Use an exchange argument.