Approximation Algorithms (Load Balancing)

July 16, 2014

Problem Definition :

We are given a set of n jobs $\{J_1, J_2, \ldots, J_n\}$.

Each job J_i has a processing time $t_i \geq 0$.

We are given m identical machines.

Problem Definition :

We are given a set of n jobs $\{J_1, J_2, \ldots, J_n\}$.

Each job J_i has a processing time $t_i \geq 0$.

We are given m identical machines.

Goal :

We want to assign (load) the jobs to machines such that the maximum load is minimized.
In other words, we would like to balance the loads.

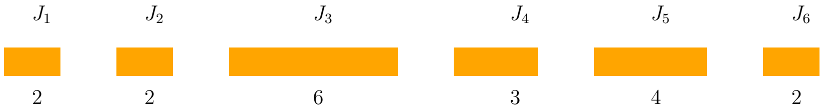
Let $A(i)$ be the set of jobs that are assigned to M_i. Then the load of M_i denoted by $T_i = \sum_{j \in A(i)} t_i$.

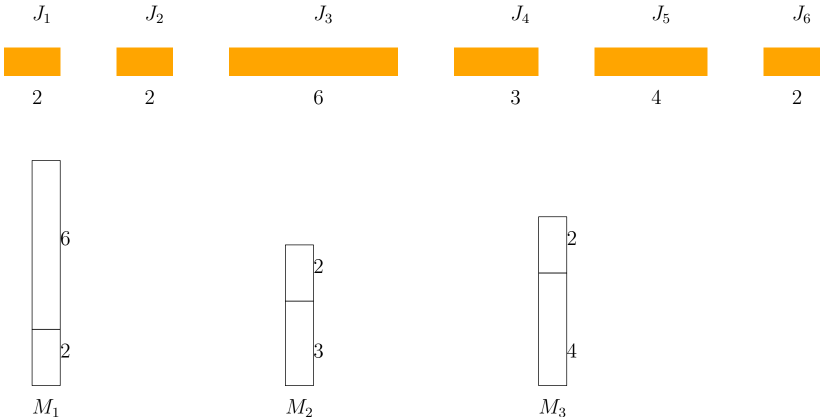
We wish to minimize $T = \max_i T_i$.

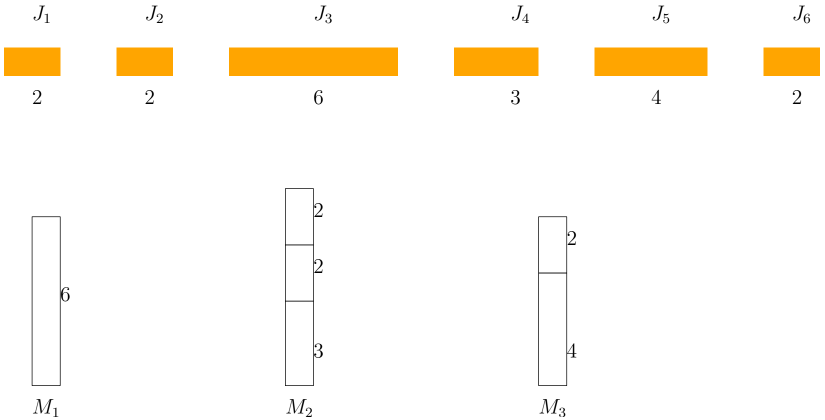
Let $A(i)$ be the set of jobs that are assigned to M_i. Then the load of M_i denoted by $T_i = \sum_{j \in A(i)} t_i$.

We wish to minimize $T = \max_i T_i$.

The load balancing problem in *NP*-complete. Even when there are two machines.


J_1
J_2
J_3
J_4
J_5
J_6
2
2
6
3
4
2


J_1
J_2
J_3
J_4
J_5
J_6
2
2
6
3
4
2
6
2
2
3
2
4
M_1
M_2
M_3


J_1
J_2
J_3
J_4
J_5
J_6
2
2
6
3
4
2
2
2
3
2
4
6
M_1
M_2
M_3

Greedy-Balance

1. Set $T_i = 0$ and $A(i) = \emptyset$ for all machines M_i.
2. **for** $j = 1$ to n
3. Let M_i be a machine with minimum load $(\min_k T_k)$.
4. Assign job j to machine M_i.
5. Set $A(i) \leftarrow A(i) \cup \{J_j\}$
6. Set $T_i \leftarrow T_i + t_j$

1) The optimal load T^* is at least

$$T^* \geq \frac{1}{m} \sum_j t_j$$

1) The optimal load T^* is at least

$$T^* \geq \frac{1}{m} \sum_j t_j$$

2) $T^* \geq \max_j t_j$.

1) The optimal load T^* is at least

$$T^* \geq \frac{1}{m} \sum_j t_j$$

2) $T^* \geq \max_j t_j$.

Lemma

Algorithm Greedy-Balance produces an assignment of jobs to machines with max load $T \leq 2T^*$.

Lemma

Algorithm Greedy-Balance produces an assignment of jobs to machines with max load $T \leq 2T^*$.

Proof.

Consider the time we add job j into machine M_i. The load of machine M_i was $T_i - t_j$ before adding J_j to M_i.

Lemma

Algorithm Greedy-Balance produces an assignment of jobs to machines with max load $T \leq 2T^*$.

Proof.

Consider the time we add job j into machine M_i. The load of machine M_i was $T_i - t_j$ before adding J_j to M_i. Also $T_i - t_j$ was the smallest load. Every other machine has load at least $T_i - t_j$. Therefore :

Lemma

Algorithm Greedy-Balance produces an assignment of jobs to machines with max load $T \leq 2T^*$.

Proof.

Consider the time we add job j into machine M_i. The load of machine M_i was $T_i - t_j$ before adding J_j to M_i. Also $T_i - t_j$ was the smallest load. Every other machine has load at least $T_i - t_j$. Therefore :

$$m(T_i - t_i) \leq \sum_k T_k$$

Lemma

Algorithm Greedy-Balance produces an assignment of jobs to machines with max load $T \leq 2T^*$.

Proof.

Consider the time we add job j into machine M_i. The load of machine M_i was $T_i - t_j$ before adding J_j to M_i. Also $T_i - t_j$ was the smallest load. Every other machine has load at least $T_i - t_j$. Therefore :

$$m(T_i - t_i) \leq \sum_k T_k$$

Also we know that $\sum_k T_k \leq \sum_j t_j$. Therefore

Lemma

Algorithm Greedy-Balance produces an assignment of jobs to machines with max load $T \leq 2T^*$.

Proof.

Consider the time we add job j into machine M_i. The load of machine M_i was $T_i - t_j$ before adding J_j to M_i. Also $T_i - t_j$ was the smallest load. Every other machine has load at least $T_i - t_j$. Therefore :

$$m(T_i - t_i) \leq \sum_k T_k$$

Also we know that $\sum_k T_k \leq \sum_j t_j$. Therefore

$T_i - t_j \leq \frac{1}{m} \sum_j t_j \leq T^*$.

Lemma

Algorithm Greedy-Balance produces an assignment of jobs to machines with max load $T \leq 2T^*$.

Proof.

Consider the time we add job j into machine M_i. The load of machine M_i was $T_i - t_j$ before adding J_j to M_i. Also $T_i - t_j$ was the smallest load. Every other machine has load at least $T_i - t_j$. Therefore :

$$m(T_i - t_i) \leq \sum_k T_k$$

Also we know that $\sum_k T_k \leq \sum_j t_j$. Therefore

$T_i - t_j \leq \frac{1}{m} \sum_j t_j \leq T^*$. Also we know that $t_j \leq T^*$.

Lemma

Algorithm Greedy-Balance produces an assignment of jobs to machines with max load $T \leq 2T^*$.

Proof.

Consider the time we add job j into machine M_i. The load of machine M_i was $T_i - t_j$ before adding J_j to M_i. Also $T_i - t_j$ was the smallest load. Every other machine has load at least $T_i - t_j$. Therefore :

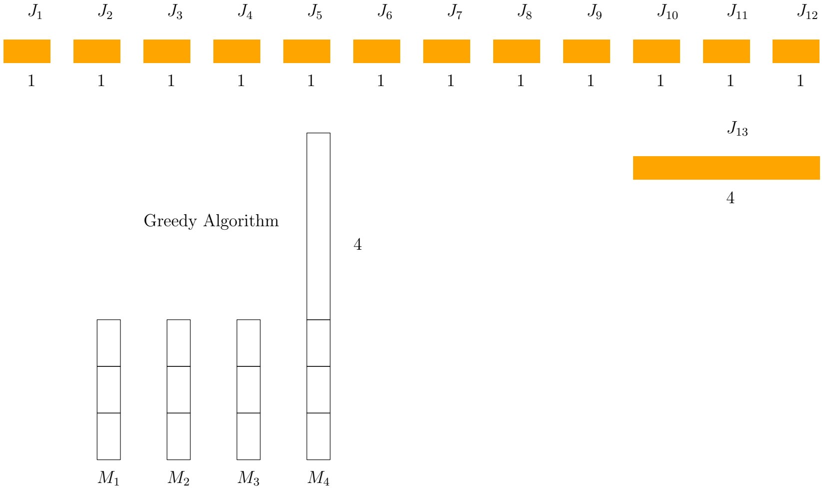
$$m(T_i - t_i) \leq \sum_k T_k$$

Also we know that $\sum_k T_k \leq \sum_j t_j$. Therefore

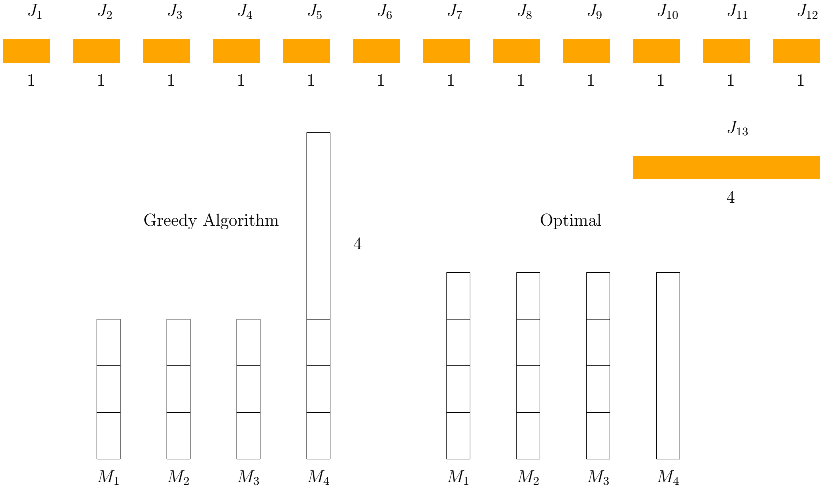
$T_i - t_j \leq \frac{1}{m} \sum_j t_j \leq T^*$. Also we know that $t_j \leq T^*$.

Therefore load of M_i after adding J_j is $T_i = (T_i - t_j) + t_j \leq 2T^*$. $\square$

The Greedy-Balance could actually be as close as possible to $2T^*$.



The Greedy-Balance could actually be as close as possible to $2T^*$.



In general suppose there are m machines and $n = m(m-1)+1$ jobs.

In general suppose there are m machines and $n = m(m-1)+1$ jobs.

The first $m(m-1)$ jobs each with time $t_j = 1$ and the last job $n = m(m-1)+1$ has time $t_n = m$.

In general suppose there are m machines and $n = m(m-1)+1$ jobs.

The first $m(m-1)$ jobs each with time $t_j = 1$ and the last job $n = m(m-1)+1$ has time $t_n = m$.

The optimal has $T^* = m$ while the Greedy algorithms has max load $2m-1$.

An Improved Approximation Algorithm

Sort-Balance

1. Set $T_0 = 0$ and $A(i) = \emptyset$ for all machines M_i.
2. Sort the jobs in decreasing order of processing times t_j.
3. Assume $t_1 \geq t_2 \geq ... \geq t_n$.
4. **for** $j = 1$ to n
5. Let M_i be a machine with minimum load ($\min_k T_k$).
6. Assign job j to machine M_i.
7. Set $A(i) \leftarrow A(i) \cup \{J_j\}$
8. Set $T_i \leftarrow T_i + t_j$

If there are more than m jobs, then $T^* \geq 2t_{m+1}$.

If there are more than m jobs, then $T^* \geq 2t_{m+1}$.

Lemma

Algorithm Sort-Balance produces an assignment of jobs to machines with max load $T \leq \frac{3}{2}T^*$.

Using similar analysis as in the previous lemma (leave it as exercise)

There exists an algorithm that find a solution for Load balancing very close to optimal T^*

There exists an algorithm that find a solution for Load balancing very close to optimal T^*

In fact there is an algorithm that for every $\epsilon > 0$ it finds a solution that is not worse that $(1 + \epsilon)T^*$.

There exists an algorithm that find a solution for Load balancing very close to optimal T^*

In fact there is an algorithm that for every $\epsilon > 0$ it finds a solution that is not worse that $(1 + \epsilon)T^*$.

But the running time of the algorithm is

$$\mathcal{O}(n^{(\frac{1}{\epsilon})^{1.5}})$$

where n is the number of jobs.