# 4

## a

We will show a bound of $O(n^4)$ steps. A step is either a move or a swap. A weight shift between machines A, B is a step that changes which machine has a larger total weight.

Lemma 1 - The number of weight shifting steps is bound by $O(n^2)$
Proof:
We have $O(n^2)$ different steps we can take - $n$ moves, and $\binom{n}{2}$ swaps. We will conclude the proof by showing that the same step cannot be taken as a weight shit twice.
Let m be a weight shift step at some point in the algorithm.
Lets assume W.L.O.G that before m machine A had a larger weight. Therefore, after m, B has a larger weight. Let L denote the weight of B before the step. Note that in because every step is improving, the weight of the larger machine (which ever it may be) decreases. Therefore, the weight of the smaller machine increases. After m is taken, the weight of the current solution is $L + weight(m)$. Let us suppose for the sake of contradiction that at some later point in the algorithm m is weight shifting again. We denote by $\tilde{L}$ the weight of the smaller machine before step m is weight shifting the second time. After step m, the weight of the solution is $\tilde{L} + weight(m)$. But $\tilde{L} > L$, so $\tilde{L} + weight(m) > L + weight(m)$ - a contradiction to the algorithm constantly improving.

Lemma 2 - the number of consecutive steps that can be made without weight shifting steps is $O(n^2)$.
Proof:
We assume without loss of generality that A is the machine with the larger weight.
For every job, let I(j) be the index of j in the order jobs by their weight, from smallest to largest.
Let $\phi$ be a potential function of A, such that Let $\phi(A) = \sum_{j \in A} I(j)$. It is easy to see every legal step decreases $\phi$.
$\phi(A)$ is at most $\sum_{i=1}^{2n} i = O(n^2)$. So the number of steps is at most $O(n^2)$.

Using the two lemmas we can see the total number of steps is $O(n^4)$.

We can also show a bound of $W/2$ -
Proof:
Let k be the difference between the two machines at the start of the algorithm.

Every we can easily see every step decreases k by at most 2 (since all weights are integer). The maximal value of k, before the beginning of the algorithm, is W. The minimal is 0. Therefore, we can have at most $W/2$ steps.

**b** Let k be the difference between the machines when the algorithm terminates. Let A be the heavier machine. $weight(A) = W/2 + k/2$. And $OPT \geq W/2$. Therefore, if $k \leq W/3$ $\frac{weight(A)}{OPT} \leq \frac{W/2+W/6}{W/2} = 4/3$.

Let $\tilde{j}$ be the lightest job in A.
Lemma 1: $weight(\tilde{j}) \geq k$
Proof:
If $k - weight(\tilde{j}) = c > 0$, by moving $\tilde{j}$ to machine B, we will improve the solution by $min(weight(\tilde{(j)}), c)$.

If $\tilde{(j)}$ is the only job in A, since $OPT \geq weight(\tilde{j})$, the algorithm is optimal. Otherwise A has another job, $j_2$. If the total weight of B is less than $weight(j_2)$, we can move $\tilde{j}$ to B, in contradiction with the assumption that the algorithm terminated.
Remember $\tilde{j}$ is the job in A with the smallest weight, so $weight(j_2) \geq weight(\tilde{j})$. Therefore, $W = weight(A) + weight(B) \geq weight(\tilde{j}) + 2 \cdot weight(j_2) \geq 3k$.
Therefore $W/3 \geq k$.

**c**

The bound is asymptomatically tight.
Example: Let n be a large even number. Let us have n+1 jobs of weight 1, and a job of weight n, and a job of weight n-1.
The optimal solution if for machine A to have n/2 jobs of weight 1, and a job of weight n, and machine B to have n/2+1 jobs of weight 1, and a job of weight n-1. This gives a solution of 1.5n. If we begin by machine A starting with the job of weight n and the job of weight n-1, and machine B starting with all the jobs of weight 1, it is easy to see the algorithm will immediately terminate. The solution in such a case will be 2n-1. $\frac{2n-1}{1.5n} \approx 4/3$.