

(a) We can assume by symmetry that the first machine  $M_1$  has the higher load. We need to prove that  $T_1 \leq 2T_2$ . Notice that our assumption that no single job takes more than half of the processing time implies the machine with higher load must have at least two jobs. Let job  $j$  be the smallest job on machine  $M_1$ . Clearly,  $T_1 \geq 2t_j$ . The local search algorithm terminated, so moving job  $j$  from machine  $M_1$  to machine  $M_2$  does not decrease the difference between the processing times. This implies that  $t_j \geq T_1 - T_2$ . We get that  $T_1 \leq t_j + T_2 \leq \frac{1}{2}T_1 + T_2$ , and multiplying by two we get that  $T_1 \leq 2T_2$ .

(b) We observed above that if a job  $j$  satisfies  $t_j \geq |T_1 - T_2|$  it cannot move. Further, the difference  $|T_1 - T_2|$  decreases throughout the algorithm, so once this condition holds, job  $j$  will never move.

Now consider a job  $j$ , and we aim to prove that  $j$  will move at most once. Assume that jobs  $j$  starts on machine  $M_1$ . So the first time it moves it will move from machine  $M_1$  to machine  $M_2$ . We always move the largest job, so at this point all remaining jobs on machine  $M_1$  will have processing time at most  $t_j$ . Consider the sequence of consecutive moves all from machine  $M_1$  to  $M_2$ , and let  $t_{j'}$  be the last job that moves in this direction (possibly  $j = j'$ ). At this time  $T_2 \geq T_1$  and  $T_2 - T_1 \leq t_{j'}$  as before moving job  $j'$  machine  $M_1$  had more work. Now we have that  $t_j \geq t_{j'} \geq |T_1 - T_2|$ , so by the observation above job  $j$  will not move again.

(c) As an example of a bad local optimum let machine  $M_1$  have two jobs with processing time 3 each. While machine  $M_2$  has two jobs with processing time 2 each. Now the difference in loads is 2, no single job can move, but swapping a pair of jobs yields a solution with identical loads.

---

<sup>1</sup>ex511.777.306