(a) Consider a progress measure Φ defined as the sum of the squares of the loads on all machines. We claim that after an improving swap move, this quantity must strictly decrease. Indeed, suppose we execute an improving swap move on machines M_1 and M_2; suppose the loads on M_i and M_j before the swap are T_i and T_j respectively, and the loads after the swap are T_i' and T_j'. Then Φ decreases by $T_i^2 + T_j^2$ and increases by $(T_i')^2 + (T_j')^2$; since $T_i + T_j = T_i' + T_j'$, and $\max(T_i', T_j') < \max(T_i, T_j)$, it follows that the decrease is larger than the increase: in other words, Φ strictly decreases.

Since there are only a finite number of ways to partition jobs across machines, Φ can only decrease a finite number of times. This implies that the algorithm terminates.

(b) Consider the machine M_i with the maximum load at the end of the algorithm. If M_i contains a single job, then since this job has to go *somewhere* in any solution, the makespan of our solution is in fact at most the optimal makespan, so our solution is optimal.

Otherwise, M_i has at least two jobs on it. We now claim that the load on M_i is at most twice the load on any other machine M_j. Since in particular this will apply to the machine M_j of minimum load, it follows also that M_i is at most twice the average, $\frac{1}{m}\sum_r t_r$, from which it follows that we are within a factor of 2 of optimal.

So suppose $T_i > 2T_j$. Since M_i has at least two jobs, the lightest job r on M_i has size $t_r \leq \frac{1}{2}T_i$. So consider the swap move in which job r simply moves to M_i. If T_i' and T_j' are the loads after this move, we have $T_i' < T_i$ and $T_j' = T_j + t_r < \frac{1}{2}T_i + \frac{1}{2}T_i = T_i$. Thus $\max(T_i', T_j') < \max(T_i, T_j)$, so this is an improving swap move, contradicting the termination of the algorithm.

[1] ex798.837.852