Scattered notes from today's lesson (Oct 26, 2022

PAOLO BOLDI

ACM Reference Format:
Paolo Boldi. 2018. Scattered notes from today's lesson (Oct 26, 2022. *J. ACM* 37, 4, Article 111 (August 2018), 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 ABOUT PTAS FOR 2-LOADBALANCING

We are hereby considering the problem of assigning tasks to two machines so that the assignment load is minimized. More formally, given a set of tasks T, each with a prescribed duration t_i (for $i \in T$), we want to divide T into two subsets, say $T_1 \subseteq T$ and $T_2 = T \setminus T_1$, so that $L = \max(L_1, L_2) = \max(\sum_{i \in T_1} t_i, \sum_{i \in T_2} t_i)$ is minimized. The pair (T_1, T_2) will be referred to as an *assignment*, L_i is the load of machine i under the assignment, and L is the assignment load.

Theorem 1.1. *Given an optimal assignment for the set for tasks X with assignment load L^*, if we further assign a new task $\bar{\imath} \in T \setminus X$ and the resulting assignment has still load L^*, then the new assignment is optimal for $T \cup \{\bar{\imath}\}$.*

Proof. Let (X_1^*, X_2^*) be an optimal assignment for X, and let L^* be the assignment load. Now, we assign the new task $\bar{\imath}$ to either machine, obtaining a new assignment (X_1, X_2) still with load L^*. Assume by contradiction that this assignment is not optimal. This means that there exists another assignment (X_1', X_2') for the set $X \cup \{\bar{\imath}\}$ such that the assignment load of (X_1', X_2') is smaller than L^*. This means

$$\max(\sum_{i \in X_1'} t_i, \sum_{i \in X_2'} t_i) < L^*.$$

Removing from X_1' or X_2' the new task $\bar{\imath}$, we obtain an assignment (X_1'', X_2'') for the set X and

$$\max(\sum_{i \in X_1''} t_i, \sum_{i \in X_2''} t_i) \leq \max(\sum_{i \in X_1'} t_i, \sum_{i \in X_2'} t_i) < L^*,$$

contradicting the fact that (X_1, X_2) was optimal for X. □

Corollary 1.2. *Assume that in one execution of the PTAS the machine with the largest load at the end is the first machine; if after the first k tasks (that are assigned in an optimal way by exhaustive search) all the remaining ones are assigned to the second machine, then the final solution is optimal.*

Proof. Theorem 1.1 says that starting from an optimal solution and assigning a new task, the new assignment remains optimal as long as the load does not change. After assigning the first k task the solution is optimal by design, and the first machine is the one with the largest load (otherwise, since the last tasks are all assigned to the second, the

Author's address: Paolo Boldi.

Manuscript submitted to ACM

second would be the most loaded also at the end). If we keep assigning the remaining $n - k$ tasks to the second machine, it is because at all steps this machine is the less loaded. This is also true also at the very end (after assigning the very last task) by the hypothesis in the statement of the theorem. Hence, the load never changes during the greedy phase, and so the final solution is also optimal. □

2 OTHER NOTES

- The fact that $P = NP$ iff $PO = NPO$ is proven in Theorem 6.2 of *Introduction to the theory of complexity* by Daniel Pierre Bovet and Pierluigi Crescenzi (see course webpage). The difficult part is of course proving that $P = NP$ implies $PO = NPO$. Under the hypothesis that $P = NP$, consider a problem $\Pi \in NPO$ (for simplicity, assume it is a maximization problem); its associated decision problem $\hat{\Pi}$ is (by definition) in NP, hence in P. So for any given input $x \in I_\Pi$ and bound k, it is decidable (in polynomial time) whether x has an admissible solution y with cost $\geq k$. With a binary search we can thus find the cost of the optimal solution $y^*(x)$. The (nontrivial) fact that from the *cost* of the optimal solution one can derive the actual optimal solution is a technical part of the proof that is based on the concept of *prefix optimization problem* (see Theorem 6.1 of the reference above).