Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
Hamid Ghodselahi January 24, 2013

Algorithm Theory, Winter Term 2012/13
Problem Set 7

hand in by Wednesday, February 6, 2013

Exercise 1: Randomized Load Balancing [8 Points]

In class, we looked at a load balancing problem, where n jobs have to be assigned to m machines such that the maximum total length of the jobs assigned to a single machine is minimized. We now look at a simple distributed version of variant of this problem. Assume all jobs have length $t_i = 1$. However, the jobs are assigned to the machines by distributed agents that cannot coordinate the assignment process. A simple strategy in such a case is to assign each job to a machine that is chosen uniformly at random (and independently for different jobs). Let X be the maximum number of jobs assigned to a single machine and let Y be the minimum number of jobs assigned to a single machine. Show that with high probability, the difference $X - Y$ is at most $O(\sqrt{n \log n})$.

Exercise 2: Load Balancing Approximation [10 Points]

In class, we considered the following load balancing problem. There are n jobs and m machines; job i requires t_i time units to be executed. The objective is to assign each job to one of the m machines such that the makespan (the maximum total processing time of a single machine) is minimized. We have seen that if the jobs are sorted such that $t_1 \geq t_2 \geq \cdots \geq t_n$, the following greedy algorithm has an approximation ratio of $3/2$. The greedy algorithm goes through the jobs in the given order and always assigns a job to the machine with the least load. In the exercise, we want to understand this algorithm a bit better.
As in the lecture, assume that T is the makespan of the solution constructed by the described algorithm and assume that i is a machine with load T in the greedy solution. Further, assume that $\hat{n}$ is the last job that is scheduled on machine i (in class, we called this job j). Note that even without jobs $\hat{n}+1, \ldots, n$, the makespan of the algorithm is T and clearly that optimal makespan when only considering jobs $1, \ldots, \hat{n}$ cannot be larger than the optimal makespan for all the jobs.

(a) Show that if an optimal solution for jobs $1, \ldots, \hat{n}$ assigns at most 2 jobs to each machine, the algorithm computes an optimal solution (i.e., $T = T^*$).

(b) Show that therefore, either $t_{\hat{n}} \leq T^*/3$ or the greedy algorithm computes an optimal solution.

(c) Based on (a) and (b), conclude that the algorithm has approximation ratio at most $4/3$.

(d) Try to find a bad input for the algorithm, where the ratio T/T^* between the makespan of the solution of the algorithm and the optimal makespan is at least a fixed constant larger than 1.
Hint: There is an example with $m = 2$ and $n = O(1)$.

Exercise 3: Online Paging Algorithms [7 Points]

In the lecture, we discussed online algorithms for paging. Assume that we have a cache of size k. We call a paging algorithm *conservative* if on any consecutive subsequence of the input containing at most k distinct page references, the algorithm will occur k or fewer page faults.

(a) Show that LRU and FIFO are conservative,

(b) Show that any conservative algorithm is k-competitive.