

Problem Set 5 – Approximation Algorithms

Due November 2

(1) Give a factor $1/2$ approximation algorithm for the following problem known as **Acyclic subgraph**.

Given a directed graph $G = (V, E)$, pick a maximum cardinality set of edges from E so that the resulting subgraph is acyclic.

Hint: Arbitrarily number the vertices and pick the bigger of the two sets, the forward-going edges and the backward-going edges. What scheme are you using for upper bounding OPT?

(2) Consider the following factor 2 approximation algorithm for the cardinality vertex cover problem. Find a depth first search tree in the given graph, G , and output the set, say S , of all the nonleaf vertices of this tree. Show that S is indeed a vertex cover for G and $|S| \leq 2 \cdot OPT$.

Hint: Show that G has a matching of size $\lceil |S|/2 \rceil$.

(3) [Kleinberg-Tardos, Ch. 11, Problem 6.]

Recall that in the basic Scheduling/Load Balancing Problem from class, we're interested in placing jobs on machines so as to minimize the *makespan* – the maximum load on any one machine. In a number of applications, it is natural to consider cases in which you have access to machines with different amounts of processing power, so that a given job may complete more quickly on one of your machines than on another. The question then becomes: How should you allocate jobs to machines in these more heterogeneous systems?

Here's a basic model that exposes these issues. Suppose you have a system that consists of m *slow* machines and k *fast* machines. The fast machines can perform twice as much work per unit time as the slow machines. Now you're given a set of n jobs; job i takes time t_i to process on a slow machine and time $\frac{1}{2}t_i$ to process on a fast machine. You want to assign each job to a machine; as before, the goal is to minimize the makespan – that is the maximum, over all machines, of the total processing time of jobs assigned to that machine.

Give a polynomial-time algorithm that produces an assignment of jobs to machines with a makespan that is at most three times the optimum.

(4) [Kleinberg-Tardos, Ch. 11, Problem 5.]

You are asked to consult for a business where clients bring in jobs each day for processing. Each job has a processing time t_i that is known when the job arrives. The company has a set of ten machines, and each job can be processed on any of these ten machines.

At the moment the business is running the simple Greedy-Balance Algorithm for the Scheduling/Load Balancing Problem that we discussed in class. They have been told that this may not be the best approximation algorithm possible, and they are wondering if they should be afraid of bad performance. However, they are reluctant to change the scheduling as they really like the simplicity of the current algorithm: jobs can be assigned to machines as soon as they arrive, without having to defer the decision until later jobs arrive.

In particular, they have heard that this algorithm can produce solutions with makespan as much as twice the minimum possible; but their experience with the algorithm has been quite good: They have been running it each day for the last month, and they have not observed it to produce a makespan more than 20 percent above the average load, $\frac{1}{10} \sum_i t_i$.

To try understanding why they don't seem to be encountering this factor-of-two behavior, you ask a bit about the kind of jobs and loads they see. You find out that the sizes of jobs range between 1 and 50, that is, $1 \leq t_i \leq 50$ for all jobs i ; and the total load $\sum_i t_i$ is quite high each day: it is always at least 3,000.

Prove that on the type of inputs the company sees, the Greedy-Balance Algorithm will always find a solution whose makespan is at most 20 percent above the average load.