Universität Bremen

FB3: Mathematik/Informatik

Prof. Nicole Megow
Dr. Syamantak Das

Summer 2017

Approximation Algorithms

Exercise Sheet 2

Exercise 1 Makespan. In the makespan minimization problem, we are given a set of identical machines $M = \{i_1, i_2, \cdots i_m\}$ and a set of jobs $J = \{j_1, j_2, \cdots j_n\}$. The processing size of job j is denotes by p_j. The objective is to determine a schedule that minimizes the maximum load on any machine or, in other words, the makespan.

Consider the following algorithm for this problem. Sort the jobs according to decreasing order of processing sizes and call this permutation J'. Now consider jobs in J' in the sorted order and assign the next job to the machine which has the least load currently. It is also known as the **LPT**(Longest Processing Time) First rule.

a) Prove that this gives a 4/3-approximation to the problem.

Let OPT denote the optimal makespan.

(i) (*5 points*) Prove that if all $p_j > \frac{1}{3}OPT$, then LPT is optimal

(ii) (*2 points*) Prove that, given an optimal solution of jobs whose sizes are bigger than $\frac{1}{3}OPT$, LPT can incur a total load of at most $4/3OPT$ on any machine after the assignment of the rest of the jobs (with processing sizes less than $1/3OPT$).

b) (*3 points*) Give a family of examples to show that the above analysis for LPT is essentially tight.

Exercise 2 Maximum k-Coverage.(*5 points*) This problem, in some sense, is a budgeted version of the set cover problem. Given a universe of elements $U = \{e_1, e_2, \cdots e_n\}$ and a family of subsets of U, $\mathcal{F} = \{S_1, S_2, \cdots S_m\}$, the goal is to select a subset $\mathcal{F}' \subseteq \mathcal{F}$ such that $|\mathcal{F}'| = k$ and $|\bigcup_{i \in \mathcal{F}'} S_i|$ is maximized.

Prove that the greedy algorithm for set cover which stops after k iterations is a $(1 - \frac{1}{e})$-approximation to the problem.

Exercise 3 Vertex Cover.(*5 points*) We discussed a 2-approximation algorithm for this problem in the lectures. However, there was another greedy algorithm suggested : Iteratively select the vertex that covers the maximum number of edges and remove all edges incident on it.

Provide an example to show that this algorithm incurs a factor strictly bigger than 2. Can you extend your example to show that the algorithm might indeed be a factor $\log n$ away from the optimal?

Exercise 4 Minimum Multiway Cut(*10 points*). This is a generalization of the minimum $s-t$ cut problem is a weighted graph. Formally, we are given a connected graph $G=(V,E)$ with non-negative weights w_e on the edges. $R=\{t_1,t_2,\cdots t_k\}\subseteq V$ is a set of terminals. The goal is to find a minimum cost subset of edges E' such that removing E' from the graph leaves all terminals in separate components.

Consider the following greedy heuristic. For $i=1,2,\cdots k$, do the following. Find the minimum cut that separates t_i from $R\setminus\{t_i\}$: let the edges be E_i. This can be done in polynomial time as follows. Contract all terminals in $R\setminus\{t_i\}$ in to a supernode S_i. All edges incident upon the terminals in $R\setminus\{t_i\}$ will now be incident on this supernode (note that this modified graph might have parallel edges). Find the minimum t_i-S_i cut in the modified graph and output that as E_i. Remove E_i from the graph, t_i from R. The final output is $E'=\bigcup_{i=1}^{k}E_i$.

Prove that the weight of E' is at most 2 times the weight of any optimal multiway cut.

(*Hint:* Observe that if you remove the edges of any optimal solution, then the graph splits in to k connected components with each t_i in a separate component)

Bonus question : Can you modify the algorithm slightly to prove that it is actually a $2-2/k$-approximation?