CSE 632: Analysis of Algorithms II Fall 2017

Homework 1

Lecturer: Shi Li **Deadline: 10/6/2017**

Instructions and Policy. Each student should write their own solutions independently and needs to indicate the names of the people you discussed a problem with. It is highly recommended that you think about each problem on your own for enough time before you discuss with others. It is recommended that you type your solutions using latex and submit the pdf file via UBLearns.

Problem 0 (0 points) These are the lemmas we did not prove in the class. You need to figure out their proofs, though you are not required to write them down.

(a) Directly show that the final k-center algorithm we give in the class (the one without using the parameter L) gives a 2-approximation.

(b) Show that the coverage function is sub-modular.

(c) (Exercise 2.2 of WS book) Show that for any input to the problem of minimizing the makespan on identical parallel machines for which the processing requirement of each job is more than one-third the optimal makespan, the longest processing time rule computes an optimal schedule.

Problem 1 (20 points, Exercise 2.1(a) of WS book) The k-suppliers problem is similar to the k-center problem we discussed. The input to the problem is a positive integer k, and a set of vertices V , along with distances $d(i, j)$ between any two vertices i, j that obey the same properties as in the k-center problem. However, now the vertices are partitioned into suppliers $F \subseteq V$ and customers $D = V \setminus F$. The goal is to find k suppliers such that the maximum distance from a supplier to a customer is minimized. In other words, we wish to find $S \subseteq F, |S| \leq k$, that minimizes $\max_{j\in D} d(j, S)$, where $d(j, S)$ is defined as $\min_{i\in S} d(j, i)$.

Give a 3-approximation algorithm for the k-suppliers problem.

Problem 2 (20 points) Let $\alpha \in [0, 1]$ be a constant. Consider the following algorithm for the maximum coverage problem:

```
1: $I \leftarrow \emptyset$
2: for $j \leftarrow 1$ to $k$ do
3:     let $i^* \in [m]$ be an index such that $\left|S_{i^*} \setminus \bigcup_{i\in I} S_i\right| \geq \alpha \cdot \max_{i'\in[m]} \left|S_{i'} \setminus \bigcup_{i\in I} S_i\right|$
4:     $I \leftarrow I \cup \{i^*\}$
5: return $I$
```

Give and prove the approximation ratio of the above algorithm.

You might think the above algorithm is only for exercise purpose, since we are "deliberately" not choosing the best i^* in each iteration. However, the above algorithm is useful in cases where the number m of sets is very large and the m sets are only given implicitly. In such a case, it may be hard for the algorithm to find a set S_{i^*} that covers the maximum number of uncovered elements; instead, it can only find a set S_{i^*} that is α-optimum.

Problem 3 (30 points) In the machine minimization problem, we are given a set $[n]$ of jobs with sizes $p_1, p_2, \cdots, p_n$ as in the makespan minimization problem we discussed in the class. We are also given a makespan T. The goal is to find the minimum integer m such that the n jobs can be scheduled on m machines with makespan T. That is, find a partition of $[n]$ into $J_1 \cup J_2 \cup \cdots \cup J_m$ with minimum m, such that $p(J_i) \leq T$ for every $i \in [m]$.

(a) Show that if we have a black-box algorithm A that solves the above machine minimization problem exactly, then we can have an algorithm A' that solves the makespan minimization problem exactly, such that A' contains polynomial number of normal operations plus polynomial number of calls of algorithm A.

(b) Give a 2-approximation algorithm for the machine minimization problem.

(c) Give a 1.5-approximation algorithm for the machine minimization problem.

Indeed, the other direction for (a) can also be proved. That means the machine minimization problem and the makespan minimization problem are equivalent, if we are looking for exact algorithms. However, we have a PTAS for makespan minimization problem, while for the machine minimization problem, there is no better than 1.5-approximation algorithm unless P = NP. That means the two problems are not the same from he approximation algorithms point of view.

Problem 4 (30 points, (Exercise 3.6 of WS book)) Suppose we are given a directed acyclic graph with specified source node s and sink node t, and each arc e has an associated cost c_e and length l_e. We are also given a length bound L. Give a fully polynomial-time approximation scheme for the problem of finding a minimum-cost path from s to t of total length at most L.