Homework 1

Please type your answers using Latex. It is allowed that you form small groups of discussion among yourselves, but everyone should write down his/her own solution. (Please also indicate with whom you have worked).

1 Scheduling on a Single Machine

Consider the following machine scheduling problem. Given a single machine, suppose that there are n jobs, each with processing time p_j, weight w_j, and a due date d_j. We schedule these jobs one by one on the machine and we gain the weight w_j of job j if its finishing time is at most d_j. We want to maximise the total weight gained.

This problem is NP-hard. But we will design an FPTAS for it in this exercise. (Do not get intimidated by this problem. You can look carefully again over how we have designed an FPTAS for knapsack problem).

Here are some hints.

1. Observe that there is an optimal schedule in which all on-time jobs finish before all late jobs.
2. Next observe we can also assume that in this optimal schedule, the on-time jobs complete in an earliest due date order.
3. Using these two observations, you should be able to solve the problem in $O(nW)$ time, where $W = \sum w_j$, using dynamic programming.
4. And then you should be able to turn this pseudo-polynomial time algorithm into an FPTAS. What is the running time?

2 K-center

We have seen in class a 2-approximation algorithm for the k-center problem.

A new trend to deal with big data is to process the input data *in parallel*: each machine receives a part of the input, does some local processing, and then sends its own output to some special machine, which then resembles the outputs of the various machines and then does a final processing.

Here is an example. Let P be the set of points in a metric space. We partition P arbitrarily into m parts $P_1 \cup \cdots \cup P_m$. Each machine i is given P_i, and then it applies

the 2-approximation algorithm to P_i to get k points $p_{i1}, \cdots p_{ik}$. These points are then sent to the special machine.

This special machine then again applies the 2-approximation algorithm to the set of points $\bigcup_{i=1}^{m}\{p_{i1}, \cdots, p_{ik}\}$ that it has received from machines 1 to m.

Prove that the final outcome is a 8-approximation for the original input P.

3 Constrained Forest

The purpose of this exercise is to make you look over again how the algorithm for Steiner forest problem that we discussed in class works.

3.1 Generalization

We say a function $f : 2^V \to \{0, 1\}$ a *proper* function if the following conditions hold.

- $f(V) = 0$,
- $f(S) = f(\overline{S})$,
- if A and B are disjoint subsets of V and $f(A \cup B) = 1$, then either $f(A) = 1$ or $f(B) = 1$.

(Check that the function f we define for Steiner forest in class satisfies these conditions).

We want to choose a subset of edges $E' \subseteq E$ with minimum cost so that for every $S \subset V$, $|\delta_{E'}(S)| \geq f(S)$. Show that the algorithm of Steiner forest can be generalized to handle this case.

To be more concrete, try to answer the following questions.

- How do you implement the "pruning step"? (Remember that in the original algorithm, we first pick a set of edges $F \subseteq E$ during the process of synchronization and then prune an edge if it is unnecessary for the feasibility of the solution).
- There is one particular lemma, which states that during the first phase, if C is a component (with respect to the currently chosen F), if $f(C) = 0$, then $\delta_{F'}(C) \neq 1$.

  Prove that your pruning step guarantees that this lemma still holds.

3.2 Point-to-Point Connection

Suppose that in the given graph $G = (V, E)$, two subsets of vertices $S, T \subseteq V$, of equal cardinality, are given, find a minimum cost subgraph so that there is a bijection $\phi : S \to T$ and vertex $s \in S$ can be connected to the vertex $\phi(s) \in T$.

Give a 2-approximation algorithm for this problem.

4 "Economical" Max-SAT

Consider the following variant of Max-SAT: given a SAT formula, where each clause C_i has a weight w_i and each variable x_j also has a weight v_j, and in particular, in each clause, all literals are positive. Here the objective is modified as follows. It is the *sum* of the weights of the clauses satisfied and the weights of the variables that are set to be false.

Write a linear program to describe this problem and then use randomized rounding (choose variable x_i to be true with the probability p_i according to your linear program) to obtain $1-1/e$ approximation. For people who like a challenge, you can try to improve the ratio further by more sophisticated rounding.