ORIE 6334 Approximation Algorithms February 13, 2014

Problem Set 2

Due Date: February 27, 2014

1. In class, we showed that a randomized greedy algorithm could be used to obtain a $\frac{1}{2}$-approximation algorithm for maximizing nonnegative, nonmonotone functions. Here we consider a deterministic variant of that algorithm, shown below. Prove that it gives a $\frac{1}{3}$-approximation algorithm for the problem, for f a nonnegative, nonmonotone function. Recall that we defined the function f over the set $\{1, \ldots, n\}$, and defined $\hat{X}_i \equiv X_i \cup \{i+1, \ldots, n\}$. For your proof, you may find it useful again to consider $\mathrm{OPT}_i = X_i \cup (\mathrm{OPT} \cap \{i+1, \ldots, n\})$, where OPT is an optimal set. Recall that for randomized algorithm we showed that

$$E[f(OPT_{i-1}) - f(OPT_i)] \leq \frac{1}{2} E[f(X_i) - f(X_{i-1}) + f(\hat{X}_i) - f(\hat{X}_{i-1})].$$

What inequality leads to a $\frac{1}{3}$-approximation algorithm?

$X_0 \leftarrow \emptyset$
for $i \leftarrow 1$ to n **do**
  $a_i \leftarrow f(X_{i-1} \cup \{i\}) - f(X_{i-1})$
  $r_i \leftarrow f(\hat{X}_{i-1} - \{i\}) - f(\hat{X}_{i-1})$
  if $a_i \geq r_i$ **then**
    $X_i \leftarrow X_{i-1} \cup \{i\}$
  else
    $X_i \leftarrow X_{i-1}$
return X_n

2. W&S Exercise 2.7

3. W&S Exercise 2.13 (a) & (b)

4. W&S Exercise 2.14

5. In the *sparsest cut problem*, we are given as input an undirected graph $G = (V, E)$ and edge costs $c_e \geq 0$ for all $e \in E$. The goal of the problem is to find a subset $S \subseteq V$ of vertices that minimizes

$$\frac{\sum_{e \in \delta(S)} c_e}{|S||V - S|},$$

where $\delta(S)$ is the set of all edges with exactly one edge in S. The sparsest cut problem tries to find a small cut in the graph such that there is a roughly equal split of the number of vertices on each side of the cut.

Show that we can find a sparsest cut in polynomial time for bounded treewidth graphs (i.e. graphs in which the treewidth k is a constant). It might be useful to know that in polynomial time one can compute a *nice* tree decomposition. In a nice tree decomposition, there are four different kinds of nodes in the tree T:

- *leaf nodes*: a leaf node i is a leaf of T and has $|X_i| = 1$;
- *introduce nodes*: an introduce node i has one child j with $X_i = X_j \cup \{u\}$ for some $u \in V$;
- *forget nodes*: a forget node i has one child j with $X_i = X_j - \{u\}$ for some $u \in V$;
- *join nodes*: a join node i has two children j and k, with $X_i = X_j = X_k$.

If the treewidth is k, then the number of nodes in T is $O(kn)$.