Problems in Approximation Algorithms

CIS800

Due: October 7th, 2010

Exercise 1. a) Prove or disprove: The greedy algorithm done in class is a $O(\log m)$ approximation for the set cover problem, where m is the number of possible sets.

b) Show that the approximation factor of the greedy algorithm described for vertex cover can be $\Omega(\log n)$, n being the number of vertices.

c) Design and analyze an approximation algorithm for the facility location problem with general connection costs.

Exercise 2. An alternate way to view the set cover problem is the following: given an $n \times m$ matrix A whose entries are in $\{0, 1\}$ with column j having cost c_j, and an n dimensional vector b all of entries are 1; pick a minimum cost set J of columns such that $\sum_{j \in J} A_{ij} \geq b_i$ for all rows i. The columns are equivalent to sets, the rows are equivalent to elements, and the set J is the set cover.

Let us consider the generalization of the above matrix problem where b is no longer an all 1's vector but can have general positive integral entries. Consider the following generalization of the greedy algorithm done in class. At iteration t, let J_t be the set of columns picked. Given column $k \notin J_t$, let the benefit of picking k be defined as

$$\mathtt{ben}(k) := \sum_{i=1}^{n} \min \left(\max(0, b_i - \sum_{j \in J_t} A_{ij}), \; A_{ik} \right).$$

(Note that for the set cover problem $\mathtt{ben}(k)$ was the number of uncovered elements in set k). Till $\sum_{j \in J_t} A_{ij} \geq b_i$ for all rows i, at each iteration add the column k which minimizes $\frac{c_k}{\mathtt{ben}(k)}$ over all k.

Show that the above algorithm is a $(\ln n)$-approximation algorithm even when the entries of b are arbitrary positive integers. Improve the analysis to $\ln k$ where k is the maximum number of 1's in any column of A. Show that the above algorithm can be as bad as a $\Omega(\max(m, n))$ approximation if the entries of A are not $\{0, 1\}$. (Thanks to Anand Bhalgat for the second problem).

Exercise 3. How many iterations do we need to implement the max-cut local search algorithm done in class? Is this polynomial time? How can you make it run in polynomial time by taking a hit in the performance? **Hint:** Move a vertex from S to $\overline{S}$ (or vice-versa) if and only if it gives *significant* increase in the cut value.

Exercise 4. a) Show that local search algorithm done in class for Max-Cut for undirected graphs is no better than a 1/2-approximation.

b) Prove that the local search algorithm for directed graphs returns a cut of weight $w(E)/4$.

c) Come up with an example where the local search algorithm for directed graphs returns a cut of weight $w(E)/4$. Come up with an example where the algorithm's cut has weight equal to $\texttt{opt}/3$.

Exercise 5. A hypergraph is a pair of sets (V, E) where each edge $e \in E$ is an arbitrary subset. This called a hyperedge to distinguish it from normal edges. A hypergraph is k-uniform if all its edges have size exactly k. Given a partition of the vertices $V = (S, \overline{S})$, the value of the cut associated is the number of hyperedges having non-empty intersection with both S and $\overline{S}$. Describe a local search algorithm for finding the largest cut in a k-uniform hypergraph. Can you analyze its performance?

Exercise 6. (*) Consider a universe U and a set value unction $f : 2^U \to \mathbb{Z}_+$. f is submodular if $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, for all subsets A, B of U. f is monotone if $f(A) \geq f(B)$ whenever $A \supseteq B$. Let $c(i)$ be the cost of $i \in U$.

Given a monotone submodular function f and a target R, the *submodular set cover* problem is to find the subset $X \subseteq U$ of minimum cost such that $f(X) \geq R$. Design and analyze an approximation algorithm for the submodular set cover problem. **Hint:** Note that submodular set cover generalizes normal set cover. (Why?)

Exercise 7. (*)

Given an undirected graph $G = (V, E)$, weights w_v on vertices, and a subset $R \subseteq V$ of terminals, the *node weighted Steiner tree problem* is to find a sub-tree of G spanning the minimum cost set of vertices containing R. Design a $O(\log |R|)$-approximation for the problem. Also show that this problem generalizes set cover.

Exercise 8. (**)

Given a graph $G = (V, E)$ with nodes $R \subseteq V$ called terminals, and $N = V \setminus R$ called Steiner. Suppose the graph is bipartite and all edges are between Steiner vertices and required vertices. Suppose $c(e)$ is the cost of edge e. The graph is *uniform* if all edges incident on a Steiner vertex have the same cost. A Steiner tree is a sub-tree of G which spans R. Consider the following algorithm for the Steiner tree problem in uniform bipartite graphs:

> *Maintain collection of connected components spanning R, initialized to singleton vertices of R. Pick a star centred at Steiner vertex v which minimizes the ratio*
>
> $$\frac{\textit{(cost of star)}}{\textit{(drop in number of connected components when star is picked)}}$$
>
> *Repeat till you get a Steiner tree.*

Show that the above algorithm achieves a 73/60 approximation, and that the analysis is tight.

Hint:

$$73/60 = \max_{k \geq 1} \frac{k + H_k}{k + 1}$$

Exercise 9. `(*)` Improve the analysis of the local search algorithm for metric facility location done in class, and show it is indeed a 3-approximation. Show that this is tight.

Hint: In the analysis in the notes, add the inequalities obtained when we state that adding any facility from X_i^* doesn't decrease cost. Also, you might have to strengthen the inequality bounding f_i (we might have to separate the clients in $\Gamma(i) \setminus \texttt{Far}(i)$ into those who get assigned to $\texttt{friend}(i)$, and those who don't).