Approximation Algorithms | **Take-home Final**
Date: Nov 30, 2005 | **Due:** Dec 12, 2005

Note: In all these problems: if we ask you to show an approximation ratio of ρ, you should tell us about the best approximation ratio you did prove, be it better or worse than ρ. Absolutely no discussion or collaboration is allowed between students when working on the exam. Finally, please refrain from consulting any external sources (papers, lecture notes, books, surveys) in attempting to solve these problems.

1. **(1,2-TSP)** Find a polynomial time approximation algorithm with performance ratio $\frac{4}{3}$ for the metric TSP problem on a complete graph where all edge costs are either 1 or 2.

   (Hint: Homework 1, Problem 4a is polynomial time solvable.)

2. **(Greedy algorithms)** The k-cut problem is defined on an undirected graph with nonnegative capacities on its edges. The goal is to find a minimum capacity set of edges whose deletion results in the graph disconnecting into at least k connected components.

   Consider the following algorithm for the minimum-k-cut problem: For every connected component of an input graph G, compute a minimum cut. Remove the lightest one. Now repeat and compute minimum cuts for all connected components. Again remove the lightest one. Repeat this procedure until there are at least k connected components. Output the union of the min-cuts that you removed.

   Show that this algorithm is a $(2 - 2/k)$ approximation for the k-cut problem.

   > (Hint: A natural way to try to do this is to "charge" the $k - 1$ cuts found by the algorithm to the k different shores of the optimal solution as we did for the multiway cut problem early in the course. However, we do not have any "terminals" in the current problem to assign identities to these shores easily.
   >
   > A very useful tool tool to carry out this charging scheme is what is called a cut-equivalent tree for the given capacitated graph. A brief description follows.
   >
   > Let $G = (V, E)$ be a connected undirected graph, and as before, let every edge $ij \in E$ be assigned a nonnegative capacity $c(ij)$. The tree we define is on the same set of vertices as G. Note that we do *not* require that the edges of T be a subset of the edges of G – the tree T is only intended to capture the structure of minimum cuts in G.
   >
   > A **cut-equivalent tree** (relative to G) is defined as a tree T with $V(T) = V(G)$ with the following property: for every edge $xy \in E(T)$, the removal of xy from T partitions $V(T) = V(G)$ into two subsets $X \ni x$ and $Y \ni y$; then we require the cut $\delta(X)$ in G to be a minimum capacity cut separating x and y.
   >
   > You may assume that you have a polynomial time algorithm available to build a cut-equivalent tree on the given capacitated graph to solve this problem.)

3. **(A Shifting Analysis)** This is not exactly something we did in class but it quite commonly used in geometric settings to design polynomial time approximation schemes. Suppose we are given n points $p_1, \ldots p_n$ in an area $I \subseteq \mathbb{R}^2$. We now want to cover all of these points with as few disks of diameter $D > 0$ as possible. In this question, you will develop a polynomial-time approximation scheme for this problem.

   (a) Suppose at first, that I is a square of side-lengths $l \cdot D$ for some constant $l > 0$. Give a polynomial-time algorithm that solves these instances exactly.
   Hint: Show first, that you need $O(l^2)$ disks to cover I. Then, prove that you need to consider only $O(n^2)$ potential positions for each disk.

(b) In the following assume that I is an arbitrary convex subset of $\mathbb{R}^2$. Look at the following natural shifting strategy: Take I and partition it along the x-axis into strips of width $l \cdot D$. Let this partition be $\Pi_1 = \{S_1^1, \ldots, S_k^1\}$. We can get another partition by *shifting* each of the S_i^1's by a distance D to the right (S_k^1 is shifted cyclically – the shifted set S_k^2 covers the leftmost strip of width D that was previously covered by S_1^1). Call this partition Π_2. Notice, that we can get l partitions $\Pi_1, \ldots, \Pi_l$ using shifts in this way and that $\Pi_{l+1} = \Pi_1$. In the following, let

$$\Pi_i = \{S_1^i, \ldots, S_k^i\}.$$

Now, suppose that you are given a ρ-approximation algorithm A that deals with instances where the length of the area I is bounded by $l \cdot D$ along one dimension. For each of the partitions Π_i, compute a solution as follows: use A to compute a feasible disk-covers $C_1^i, \ldots, C_k^i$ for each of the strips $S_1^i, \ldots, S_k^i$. Then let $\mathtt{APX}^i = C_1^i \cup \ldots \cup C_k^i$. Return

$$\mathtt{APX} = \mathrm{argmin}_{1 \le i \le l} |\mathtt{APX}^i|.$$

Show that $|\mathtt{APX}| \le \rho \cdot (1 + 1/l)|\mathsf{OPT}|$.

(c) How do (a) and (b) lead to a PTAS for the euclidian disk cover problem?

4. **(Densest Subgraph)** Given an undirected graph $G = (V, E)$, the density of a subset S is defined to be

$$\rho(S) = \frac{|E(S,S)|}{|S|}; \tag{1}$$

i.e., the number of edges with both endpoints in S, divided by the number of vertices in S. The goal of this problem is to find the subset S which maximizes $\rho(S)$, the max being taken over all subsets of V.

(a) Consider the following LP relaxation for the problem: have a variable x_{ij} for each edge $\{i, j\}$, and variable y_i for each vertex $i \in V$. The goal is to maximize $\sum_{ij} x_{ij}$ subject to the constraints:

$$\begin{aligned} x_{ij} &\le \min\{y_i, y_j\} \qquad\qquad \forall \{i,j\} \in E \\ \textstyle\sum_{i \in V} y_i &\le 1. \end{aligned}$$

Is this a *linear* program? If not, write an equivalent LP for the problem. Show that this is a valid relaxation for the Densest Subgraph problem.

(b) Give the best (upper and lower) bounds you can on the integrality gap of this LP relaxation.

5. **(Local Ratio is Primal Dual)** Recall from the lectures on the Local Ratio method that the process of "stripping off" one edge at a time to build a decomposition yields a 2-approximation algorithm for the vertex cover problem (cf. Lecture 9 notes).

Such local ratio decomposition steps naturally lead to an idea for primal-dual algorithms, esp. in terms of the dual update step. In particular, this naturally defines a primal-dual algorithm for the vertex cover problem where at every step, we pick any (as yet uncovered) edge and raise its dual value until the dual constraint corresponding to either endpoint becomes tight. Use this ideas to write down a primal dual algorithm for vertex cover and show how it's performance ratio is 2.

Then, do the same for the directed spanning tree problem from Homework 4, Problem 1. Be very careful here: First write the ILP formulation for the directed spanning tree problem with an exponential number of constraints (similar to the one for undirected Steiner trees that we wrote in class). Then show how the local decomposition steps in the algorithm can be used to define an appropriate dual update step. Finally, show that this approximation ratio has performance ratio 1 (i.e., it is an exact primal-dual algorithm for directed spanning trees)!

6. **(Unique Set Cover)** Consider the following problem: we are given a universe U with n elements, and a family of sets $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$. The goal is to pick some subfamily $\mathcal{F}'$ of K sets to maximize the number of elements that are covered by *exactly one* set in this subfamily. (I.e., they are covered *uniquely*.)

   (a) Suppose we are working in a simple case when all the elements of U are contained in exactly d of the sets in $\mathcal{F}$. Choose a probability $p = p(d)$ such that adding each set from $\mathcal{F}$ to our family $\mathcal{F}'$ with probability p will result in $\Omega(1)$ of the elements of U being covered uniquely.

   How does your analysis change if each element u is contained in d_u of the sets, where $\forall u \in U, d_u \in [d, 2d]$ for some universal quantity d?

   (b) By classifying the elements into suitable groups and using the above algorithm, obtain an $O(\log D)$ approximation algorithm for the problem, where D is the maximum number of sets in which any element in U is contained. Show that this implies an $O(\log m)$ approximation algorithm for the problem.

   (c) (Not very difficult, but not for credit) Can you extend this idea to get an $O(\log n)$ approximation algorithm for the problem?