CS 880: Approximation Algorithms — Homework 1

Out: 2/01/07 — **Due: 2/22/07**

1. Approximating the TSP on directed graphs is much harder than on undirected graphs. The best known algorithm for the former achieves an $O(\log n)$-approximation. In this question we will consider the TSP in special directed graphs that we will call $\{1, 2\}$**-graphs**. These are complete graphs with each directed edge of length 1 or 2. (Convince yourself that these lengths satisfy the triangle inequality.)

   (a) First we will look at a problem closely related to the TSP. In the **cycle-cover problem** our goal is to find a minimum weight collection of simple (directed) cycles in the graph such that each vertex in the graph is contained in exactly one cycle. Prove that the minimum weight cycle cover in a graph can be found in polynomial time. (Cycles of length 2 are allowed.)

   (b) Use the algorithm from part (a) to give a $3/2$-approximation for the TSP on $\{1, 2\}$-graphs.

   (c) Can you improve your algorithm from part (b) to obtain a $4/3$-approximation? What property do you require from the cycle cover algorithm in order to obtain this improvement?

2. In the k**-cut problem**, we are given a weighted graph, and asked to return a cut of minimum weight that separates the graph into at least k connected components. Note that this problem is very similar to min multiway cut, except that we are not given specific terminals to separate.

   Consider the following algorithm for k-cut: at every step, find the min-cut in every component in the graph, remove the smallest weight min-cut, and repeat until k components are formed.

   Prove that this algorithm achieves a $2(1 - 1/k)$-approximation to k-cut.

3. Recall that in the set cover problem our goal is to cover a given set of elements using a collection of subsets of the least total cost. In the **maximum coverage problem** our goal is complementary—we want to cover the most number of elements using a collection of subsets with total cost no larger than a given bound B.

   (a) Give a constant factor approximation for the maximum coverage problem. Try to obtain as good an approximation as possible.

   (b) Prove that maximum coverage cannot be approximated within a factor of $1 - 1/e + \epsilon$ for any constant $\epsilon > 0$ unless NP$\subset$ DTIME$(n^{\log \log n})$. *(Hint: Use Feige's hardness result for set cover.)*

4. Recall the min makespan scheduling algorithm from class, and consider the following variant of the algorithm—schedule jobs in decreasing order of size and send each job to the shortest queue so far.

   (a) Prove that this algorithm obtains a $3/2$-approximation.

   (b) Give the worst gap example you can think of for this algorithm.

   (c) (*This part is not for credit.*) Can you prove that the algorithm achieves a $4/3$-approximation?

5. A **legal** k**-coloring** of a graph is an assignment of colors $1, 2, \cdots, k$ to the vertices of the graph such that no two adjacent vertices receive the same color. A graph is k**-colorable** if there exists a legal k-coloring of its vertices. The problem of finding a legal k-coloring of a k-colorable graph is NP-complete for $k \geq 3$.

   (a) Prove that graphs with maximum degree Δ are $(\Delta + 1)$-colorable. Also give a polynomial time algorithm for finding a $(\Delta + 1)$-coloring.

   (b) Give a polynomial time algorithm for 2-coloring a bipartite graph.

   (c) Using parts (a) and (b) above, give a polynomial time algorithm for finding an $O(\sqrt{n})$-coloring of a 3-colorable graph.

   (Hint: Verify and use the fact that the neighborhood of any vertex in a 3*-colorable graph is* 2*-colorable.)*

(d) Extend the algorithm from part (c) to obtain an $O(n^{2/3})$-coloring for a 4-colorable graph in polynomial time.

(Aside: The best known algorithm for coloring 3-colorable graphs uses $O(n^{0.2111})$ colors.)

6. In the **Unique Set Cover** problem, we are given a set of n elements and a collection of m subsets of the elements. Our goal is to pick out a number of subsets so as to maximize the number of **uniquely covered** elements, those that are contained in exactly one of the picked subsets. (Note that the cost or number of subsets picked is not important).

   (a) Consider the following naïve algorithm—for some number $p < 1$, the algorithm picks each subset independently with probability p. Assuming that every element is contained in exactly F subsets, compute the expected number of elements uniquely covered. For what value of p is this expectation maximized?

   (b) Assuming that each element is contained in at most F subsets and at least $F/2$ subsets, give a constant factor approximation to unique set cover using the algorithm from part (a).

   (c) Extend the algorithm from part (b) to obtain an $O(\log m)$ approximation in general (without assumptions on the frequency of any element). *(Hint: Try reducing this problem to the one in part (b).)*

   (d) Can you improve the approximation from part (c) to a factor of $O(\log n)$?
   (Hint: Can you limit the number of sets under consideration to only n?)