Due Wednesday, June 20

Reading: 1. PDF file on Probabilistic Algorithms (posted on the class home page)

Problems: 10 points each.

1. i. Consider the 2 approximation algorithm for VC that we discussed in class, and can be found on page 1110 of the textbook.

   Find and define an infinite collection of graphs $G_n$ for infinitely many $n$, such that $G_n$ has at least $n$ vertices and the algorithm produces a VC for $G_n$ which is exactly 2 times the size of the smallest VC for $G_n$.

   Since we proved that the algorithm always produces a VC which is no more the twice the optimal VC value, this example shows that the analysis of that algorithm showing it is a 2 approximation for VC cannot be improved to get it to be a k approximation algorithm for some k less than 2.

   ii. Now consider the heuristic algorithm for the VC problem where at each step you choose a vertex $v$ in the graph of highest degree. You then delete $v$ and all edges incident to $v$ from $G$. You repeat this until all edges have been deleted from the graph.

   Find an example graph $G$ where this algorithm does not result in the optimal VC being found. Try to make your example as far from optimal as possible. (Making it more than 1.5 time the optimal would be very good.)

2. Find an example of a traveling salesman problem instance where the 1.5 approximation algorithm is not optimal.

3. Find an example of a weighted graph $G$ which is an instance of the minimum weighted VC problem with the property that when you write down the IP for this graph and then relax it to an LP, the LP has a solution that is not all integers. What VC do you get for your graph when you round the solution values to integers?

   Your graph should have at least 3 variables and 2 edges, but try to keep it small as otherwise it may be hard to find the optimal LP solution.

4. Recall the max-cut problem discussed in class.

   Here is an approximation algorithm $A$ for finding a cut $(S,T)$ of a graph $G$ which is within a factor of two of the size of $G$’s maximum cut.

   Algorithm $A$:
   1. Initially put all vertices into $S$ and none in $T$.

   2. Run through every vertex $v$ in $G$. For each $v$ move $v$ to the other side of the cut if this increases the size of the cut. If not, leave it where it is.
3. If step 2 caused some vertex to move then repeat 1. Else halt.

Prove that A gives a 2-approximation for max cut.

What is A’s running time?

5. Give an example of a IP where the best LP solution of the relaxation of the IP is more than 20 times better than the best IP solution in terms of the value of the objective function values obtained.