CS 530 - Summer 2018
Homework 4 - Brief answers to problems 1 and 2

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.

Answer:

Let $G$ be the edge v1—v2 replicated n times. The min VS is one bvertex from each edge. So of size n. The approx alg put all the 2n vertices into the approximate cover, so has size 2n.

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.)

Just to get non-optimal is pretty straightforward, but to get close to 2 times optimal is difficult.

Consider 3 copies of of the segments a—b—c, all three joined at their “a” nodes. Then the optimal VC will be the 3 middle (b) nodes and the approximation algorithm will choose a and three other nodes, so consist of 4 nodes. This is not optimal.

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

Answer:

Consider the complete graph which on 4 vertices (A square) where the 4 edges around the square have weights 4,3,4,2 and the diagonals have weights 7 (from top left to bottom right) and 2 (from bottom left to top right).

Rough picture (without the 2 diagonal drawn in):

```
  4---7---2  3---2---4
  ^         |     |
  |         |     |
  |         |     |
  3         7     1
```

The the MWST consists of the 2 side edges (2 and 3) and the diagonal edge of weight 2. We then add in the matching of the two odd degree vertices in the MWST which is the diagonal of weight 7. This forms and euler cycle of weight $2+3+2+7 = 14$ which our 1.5 approx alg gives us.

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
Let $G =$

\[
\begin{align*}
& a, w = 5 \\
& b, w = 10 \\
& c, w = 6
\end{align*}
\]

The IP corresponding to the weighted VC problem for $G$ is:

\[
\text{minimize} \quad 5x_a + 10x_b + 6x_c \\
\text{subject to} \\
x_a + x_b \geq 1 \\
x_b + x_c \geq 1 \\
x_a + x_c \geq 1 \\
x_{a1}, x_{b1}, x_{c1} \leq 1 \\
x_{a1}, x_{b1}, x_{c1} \in \mathbb{N}
\]

The solution to the relaxation of this IP to a LP gives $x_a = .5, x_b = .5, x_c = 5$ where the objective function has value 10.5. The VC this would correspond to is $\{x_{a1}, x_{b1}, x_{c1}\}$ which actually has weight 21.

4. $A$ is a 2-approximation:

The size of the max-cut of any graph $G$ is bounded by $|E|$, the number of edges in $G$, since the max-cut could at most have every edge crossing it. So, to prove that $A$ is a 2-approximation, we prove that the cut that $A$ produces has at least $|E|/2$ edges.

Suppose not, so suppose that $A$ terminates and produces the cut $(S, T)$ which has less than $|E|/2$ edges crossing it. Let $d(v)$ be the degree of vertex $v$. There must be some vertex $v$ for which more than $d(v)/2$ of $v$'s neighbors are on the same side of the cut as $v$ since otherwise the size of $(S, T)$ would be at least $|E|/2$. Switching vertex $v$ to the other side of the cut improves the cut since now more than $d(v)/2$ edges cross the cut and before fewer did. This is a contradiction because $A$ would not have terminated with solution $(S, T)$ if moving $v$ to the other side of the cut improves the cut. Thus, $A$ must produce a cut with size at least $|E|/2$.

Running time of $A$:

The initialization of the algorithm just puts all of the vertices into $S$, which takes $O(|V|)$ steps.

The algorithm starts with a cut of size 0 (all vertices in $S$) and in each iteration it improves the size of the cut by at least 1 (besides the very last iteration). Therefore, the number of iterations is bounded by the size of the max cut which can be at most $|E|$.
Each iteration checks for each vertex whether moving the vertex to the other side increases the size of the cut. To check if moving \( v \) increases the size it suffices to calculate how many neighbors \( v \) has on the same side and how many it has on the opposite side. This can be done in \( O(|E|) \) steps. Moving \( v \) can be completed in constant time so in total, each iteration requires \( O(|V||E|) \) steps.

In total, the running time of \( A \) is \( O(|V||E|) \).

Consider the IP below.

\[
\text{maximize } 50x + y
\]

subject to

\[
\begin{align*}
2x &\leq 1 \\
y &\leq 1 \\
x &\in \mathbb{N} \\
y &\in \mathbb{N}
\end{align*}
\]

The solution to the IP is \( x = 0, y = 1 \) which gives the objective function a value of 1. The solution to the relaxation of the IP to a LP is \( x = \frac{1}{2}, y = 1 \) which gives the objective function a value of 25, which is more than 20 times larger than the value of the objective function for the IP.