UTAH STATE UNIVERSITY, COMPUTER SCIENCE DEPARTMENT

CS 7910 Computational Complexity
Assignment 10

Gopal Menon

April 23, 2016

1. **(20 points)** In this exercise, we design an approximation algorithm for the dominating set problem. We have proved in class that the dominating set problem is NP-Complete. Here we consider its optimization problem.

   Given an undirected graph G of n vertices, a subset S of vertices of G is a dominating set if each vertex v of G is either in S or connects to a vertex of S by an edge. The problem is to find a dominating set of G of minimum size.

   Design a polynomial-time approximation algorithm for the problem with approximation ratio $O(logn)$. In other words, if OPT is the size of the optimal dominating set and C is the size of the dominating set found by your algorithm, then it should hold that $C \leq O(logn) \cdot OPT$, which is equivalent to $C = O(OPT \cdot logn)$ by the definition of the big-O notation.

   Consider the following approximate algorithm for finding a dominating set, that takes a graph G having a set of V vertices and E edges as an input parameter

   Algorithm 1 Greedy Dominating Set Approximation Algorithm

   procedure GREEDY DOMINATING SET APPROXIMATION ALGORITHM(G)
       Create remaining vertices set $R = V$
       Create an empty set of vertices $S = \phi$
       while $R \neq \phi$ **do**
           Select vertex v from R such that set S_i consisting of vertex v
           and all vertices connected to v by an edge maximizes $S_i \cap R$
           $R = R - S_i$
           $S = S \cup v$
       end while
       return S
   end procedure

   For every element $x \in S_i \cap R$ in algorithm 1, we can associate a cost $C_x = \frac{1}{S_i \cap R}$. If we add up the cost for every element in set S_i we will get a total of 1. The size of the dominating set found by this algorithm will be the number of vertices in set S. Since the dominating set will cover all vertices either by including them in S or by having an edge to it from a vertex in set S, the number of vertices C in the dominating set will be given by

$$C = \sum_{x \in V} C_x$$

We need to show that this algorithm will give us a dominating set of size

$$C \leq O(logn) \cdot OPT$$

where OPT is the number of vertices in the optimal dominating set.

We will start by proving the following lemma for any subset of vertices S_k

$$\sum_{x \in S_k} C_x \leq H(d_k) \tag{1}$$

where H is the harmonic function defined as

$$H(n) = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n} = \theta(logn) \tag{2}$$

and

$$d_k = |S_k|$$

Let

$$S_k = \{x_1, x_2, \ldots, x_{d_k}\}$$

be the set S_k with its elements ordered by the time they are first removed from the set of remaining vertices R. Consider an element x_j of S_k. Suppose x_j is first removed in an iteration when we remove S_i from R. $x_j \in S_i \cap R$ at the beginning of the iteration. And $x_j, x_{j+1}, \ldots, x_{d_k}$ have not been removed from R yet. Since the set S_i may or may not be the same as S_k,

$$|S_i \cap R| \geq |S_k \cap R| \geq d_k - j + 1$$

We can see that

$$C_{x_j} = \frac{1}{S_i \cap R} \leq \frac{1}{d_k - j + 1}$$

for any $1 \leq j \leq d_k$.
So

$$C_{x_1} \leq \frac{1}{d_k}, C_{x_2} \leq \frac{1}{d_k - 1}, \ldots, C_{x_{d_k - 1}} \leq \frac{1}{2}, C_{x_{d_k}} \leq 1 \tag{3}$$

Adding everything in 3, we get

$$\sum_{x \in S_k} C_x \leq 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{dk - 1} + \frac{1}{d_k} = H(d_k) \tag{4}$$

So we can see from 4 that 1 is proved. Let A^* be the set of vertex sets formed by the union of all sets with each set consisting of a vertex in the optimal solution along with all the vertices connected to it by edges in the graph G. For the size C of the dominating set found by the algorithm

$$C = \sum_{x \in V} C_x \leq \sum_{S_k \in A^*} \sum_{x \in S_k} C_x \tag{5}$$

Using 1, 5 reduces to

$$\begin{aligned}
C &\leq \sum_{S_k \in A^*} H(d_k) \leq \sum_{S_k \in A^*} H(d_{max}) \\
C &\leq H(d_{max}) \cdot \left| A^* \right| \\
C &\leq O(log(d_{max})) \cdot OPT
\end{aligned}$$

Since d_{max} (the size of the largest set of a vertex and all vertices connected to it by edges) is limited by the number of vertices n in graph G,

$$C \leq O(log(n)) \cdot OPT$$

$$C = O(OPT \cdot log(n)) \tag{6}$$

Equation 6 proves the requirement.

2. In this exercise, we consider a "dual" problem of the load balancing problem.

   Suppose there are m machines and n jobs such that each job i has a processing time t_i. Consider a job assignment that assigns each job to one of these machines. For each machine j, let T_j denote the total sum of the processing time of all jobs assigned to machine j, and we call T_j the workload of machine j. We call the value $min_{1 \leq j \leq m} T_j$ the *minimum workload* of all machines of the assignment.

   The *dual load balancing problem* is to compute a job assignment that *maximizes* the minimum workload of all machines.

   Remark. Recall that the load balancing problem is to find a job assignment that minimizes the maximum workload of all machines. Therefore, the two problem are "dual" to each other.

   a) **(5 points)** The dual load balancing problem defined above is an optimization problem. What is the decision version of this problem?

   b) **(10 points)** Prove that the decision problem is NP-Complete.

   c) **(15 points)** Let A be the sum of the processing time of all jobs, i.e., $A = \sum_{i=1}^{n} t_i$. We assume that $t_i \leq \frac{A}{2m}$ for each job i (intuitively, each t_i is not "too big"). Under this assumption, design a polynomial-time approximation algorithm for the problem with approximation ratio 2. In other words, if OPT is the minimum workload in an optimal solution and C is the minimum workload in your solution, then it holds that $C \geq \frac{1}{2} \cdot OPT$.