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.

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 $C \geq \frac{1}{2} \cdot OPT$.