CS 6505 - Homework 12

Caitlin Beecham

3

We reduce 3-SAT' to Problem P2*.

(a) Form a graph from the clauses in 3-SAT' by doing the following

  (i) For each clause, add a new vertex for each literal in the clause (boolean variables and their negations may well appear multiple times across the whole graph we are constructing) and add an edge between each vertex representing a literal in this clause (forming a triangle). Repeat for all clauses.

(b) Then, for each boolean variable, x_i

  (a) For each instance, v_j, of x_i as a vertex (it may appear many times):

  - For each instance, w_k, of $\bar{x_i}$ as a vertex,
    - Connect v_j and w_k with 3K+1 multiple edges (where K is the number of clauses). (This construction ensures that when we run our algorithm to maximize the number of edges between our two partition sets x_i and $\bar{x_i}$ will always appear on opposite sides of our partition (each of which corresponds to TRUE or FALSE))

(c) We have now finished constructing G. Run our algorithm to find a partition of G into 2 non-empty sets such that the number of edges between them is maximized. If the output of this partition results in a structure where, for each clause (represented by a triangle), some vertices are in one partition set and some are in the other, then it is possible to find an assignment to 3SAT'. Otherwise, if the outputted structure has one triangle corresponding to some clause completely contained in one of the partition sets, then no assignment solves the 3SAT' problem.

Problem P2: Given a graph G = (V, E), partition V into two non-empty subsets, maximizing the number of edges between the part.
Thus, since we have reduced, 3-SAT', an NP complete problem to Problem P2, this means that P2 is NP-complete and thus is not polynomial time solvable unless P=NP.

Next we show that P1 is at least as difficult as all NP complete problems (I don't want to say it is NP-complete since technically it's not a decision problem). Namely, we reduce P2 to P1, meaning that we solve P2 using an algorithm for solving P1. So, say we have some multigraph G=(V,E), given by the adjacency matrix a^i_j = number of edges between vertex i and vertex j Define $K = \max_{i,j}\{a^i_j\}$. Then, define a new graph G' by the following adjacency matrix, b^i_j. We define $b^i_j := K - a^i_j$. Then, we run the algorithm for P1 on G'. This will return a partition of G' into two non-empty subsets such that the number of edges between the partition sets is minimized. By construction, such a partition S_1, S_2 such that $S_1 \cup S_2 = [n]$ and $S_1 \cap S_2 = \emptyset$ is one that maximizes the number of edges across the partition in the ORIGINAl graph G. Thus, we have reduced P2 to P1, in polynomial time. Thus, P1 is at least as difficult as NP-complete problems.