CS 341, Fall 1999 due Friday Dec. 3

A. Lubiw, D. Schuurmans

ASSIGNMENT 5 SOLUTIONS

These are solution sketches, not model solutions.

1. A problem is in co-NP if there is a polynomial time verification algorithm for its NO instances. A Boolean formula is a NO instance of the Tautology problem if there is some assignment of 0-1 truth values to the variables that makes the formula not true. Given as a certificate such an assignment of 0-1 values to the variables, one can check in polynomial time that the formula evaluates to false. Thus Tautology is in co-NP.

2. Suppose we have a polynomial time algorithm $\mathcal{A}$ for 3-SAT. Given a 3-CNF formula F we would like to find a satisfying truth-value assignment. First, use $\mathcal{A}$ to check if F is satisfiable. Next, take a variable x that appears in F and set it to True. Simplify F. Note that the resulting formula F' will still be 3-CNF. Use $\mathcal{A}$ to check if F' is satisfiable. If it is, then we can keep x = True, and continue to find a satisfying truth-value assignment for the variables of F'. On the other hand, if F' is not satisfiable, then we must set x = False, simplify F to F'', and continue to find a satisfying truth-value assignment for the variables of F''. The number of times we repeat this step is equal to the number of variables. The simplification we do in each step takes time proportional to the number of clauses. Thus the whole algorithm takes polynomial time.

3. First note that this problem is in NP—a certificate is a simple path from s to t of weight at most k, and checking such a certificate can be done in polynomial time. Next, we reduce Hamiltonian Cycle to this problem. Given an undirected graph G as input to Hamiltonian Cycle, make a directed graph G' by putting each edge of G in each direction. Put a weight of -1 on each edge. Choose an arbitrary vertex v of G and set $s = t = v$. Set $k = -(n-1)$. This reduction can be done in polynomial time. Now prove that G has a Hamiltonian path iff G' has a simple path from s to t of weight $\leq k$. [Note: this assumes that $s = t$ is allowed for the simple path problem. It is OK to make this assumption, but you can do without it by splitting v into two vertices v_{in} and v_{out}, and attaching all the edges leaving v to v_{out}, and all the edges entering v to v_{in}. Then set $s = v_{\text{out}}$ and $t = v_{\text{in}}$.]

4. First note that the problem is in NP. Next, reduce Subset Sum to this partitioning problem. Suppose we have an algorithm for the partitioning problem. Suppose we are given a set S of numbers and a target t as input to the Subset Sum problem. Let s be the sum of all elements in S. If $t = s/2$ then the algorithm for the partitioning problem already solves the Subset Sum instance, since there is a subset of sum t iff the set can be partitioned into two subsets with equal sum.
   If $t > s/2$ consider adding one element $e = 2t - s$ to S. We claim that $S \cup e$ has a partition into two sets of equal sum iff S has a subset of size t. (Prove this.) This provides a valid reduction

so long as we allow multiple elements in our set. (The point is that $2t - s$ might already be in S.) One way to eliminate such a pair of duplicate elements, e and e', is to multiply all other elements of S by 5, and throw in $5e - 1$, $5e' - 2$, 1, and 2. Verify that the resulting set has distinct elements and that it can be partitioned into two subsets of equal sum iff the original multiset can. This reduction takes polynomial time.
Finally, if $t < s/2$, add $e = s - 2t$, and proceed as above.

5. Observe first that in a cycle of $6k$ vertices—which has a vertex cover C of size $3k$ consisting of every other vertex in the cycle—the heuristic can choose every third vertex (they all have degree 2), leaving $2k$ loose edges that need to be covered by $2k$ vertices. Thus the heuristic gives a vertex cover of size $4k$, for a ratio of $4/3$. Now, if we add $3k - 2$ vertices adjacent to all the vertices in the minimum cover C, this doesn't change the minimum cover, but now these new vertices all have degree $3k$, the vertices of C have degree $2 + 3k - 2 = 3k$, and the remaining vertices have degree 2, so the heuristic can pick all the new vertices first, and then the remaining graph is the cycle. Thus the heuristic gives $3k - 2 + 4k$ vertices, for a ratio of $(7k - 2)/3k$. This is greater than 2 so long as $k \geq 3$. The smallest example built this way has 25 vertices.

6. The problem is solvable in polynomial time.