Theory of Algorithms. Homework 6

Peter Racioppo

1.) Monotone Satisfiability with Few True Variables

Given a monotone instance of satisfiability, we are asked whether there is a satisfying assignment for the instance in which at most k variables are set to 1. We first need to show that the problem is in NP. We can verify a satisfying assignment simply by plugging in assigned variables to each clause. Whether each clause is satisfied can be checked in polynomial time and assuming there are a polynomial number of clauses, the total verification is polynomial in the number of variables.

We now need to show that the problem is NP-complete. Any 3-SAT problem can be expressed as a k-SAT problem by adding dummy clauses, so 3-SAT reduces to k-SAT. For instance, AvBvC —> AvBvC^(DvD'), which is true if and only if AvBvC is true. 3-SAT is NP-complete, so k-SAT is as well.

Alternatively, we can reduce SAT to k-SAT. This can be done in the same way that SAT is reduced to 3-SAT: for a particular clause AvBvCvD, we can write an equisatisfiable clause (AvBvC)^(C'vD). We can always split a clause of length k into two clauses of length k-1 in this way, and doing this iteratively takes us from clauses of any length to any lower length after no more than k steps. Thus, SAT reduces to k-SAT, which reduces to 3-SAT.

2.) The Strategic Advertising Problem

a.) We need to show that the problem is NP-complete. Vertex cover should probably reduce to strategic advertisement.

The problem is in NP because there are polynomial edges in a complete graph and thus polynomial paths, so it would only take polynomial time to verify whether a given set of vertices covers the full set of paths.

Suppose that we know the vertex cover for some undirected graph. A residual directed graph can be built from this graph by directing all the edges incident on the vertices in the vertex cover either in or out of these vertices. If an edge is incident on more than one vertex in the vertex cover, remove it from the residual graph. By construction, every path in the residual graph has length one. This is because moving through the graph requires us to alternate between covering and non-covering vertices, and every such path is alternating in terms of edge directions. Also, a vertex cover for any graph also covers the graph formed by removing the edges between vertices in the vertex cover. Now, since all of the paths in the residual graph have length one, finding the minimal covering of the paths is equivalent to finding the vertex cover. Thus, finding the minimal path covering in the residual graph is equivalent to finding the vertex cover in the original graph.

In summary, we've established by construction that given an undirected graph, there always exists a directed residual graph such that finding the minimal covering of its paths gives us the vertex cover of the original graph. Thus, vertex cover reduces to strategic advertisement, which implies that the latter is NP complete.

b.)

We can use the black box algorithm to test whether or not a particular vertex is part of the minimal cover. To start, run the black box to determine that there are k vertices in the minimal cover. Now, choose an arbitrary vertex and remove all of the vertices that are connected to it by a single directed path. Run the black box again on the part of the graph that was not removed. If the number of vertices in the minimal cover for this subset has decreased by one, the previously tested vertex is part of the solution; otherwise, it is not. This is the case because if this vertex is not part of the solution, every vertex reachable from it will also be reachable from some other vertex in the cover, and so removing this section of the graph will not decrease the cardinality of the solution. The number of vertices in subsets can decrease by no more than one between steps by construction. The algorithm is clearly polynomial in the number of vertices, because we can eliminate at least one vertex from consideration after each step. Each step involves only

the polynomial invocation of the black box and a DFS or BFS to find connected elements. Run time can be further decreased (possibly) by limiting the search to vertices that are on roots of arborescences or connection points between them. I believe representing the graph in terms of arborescences also provides a means of reducing the problem to a vertex cover.

3.) Canisters in Trucks

a.) The canisters and the k spots in the trucks can be represented as a bipartite graph, with edges representing safe assignments. Use Hopcroft-Karp to find a maximum cardinality matching.

b.) Assign a vertex to each canister. Begin with a fully connected graph, and then remove the edges of the canister pairs that are not permitted. The problem is to find the maximal number of k-sized groups of connected vertices.

The problem is in NP since, given a set of k-sized groups, we can check whether there are violations in time linear with the number of vertices by, for each k-sized group, moving through the vertices one by one and setting an indicator variable for each type of canister. There are no more indicator types than number of vertices, so at each of the vertices we perform a polynomial time operation. We can also check whether every element is included exactly once in linear time (wrt vertices).

We can probably reduce set-cover to the group-canisters problem. Suppose that for a set U of elements, we enumerate all of the possible sets $S=\{S1,\ldots,Sz\}$ that are of cardinality less than or equal to k and that obey the pairwise constraints. Our job is to find a collection of no more than m sets in S whose union is U. The group-canisters problem is a restriction of set-cover, in that we impose the additional constraint that no two sets in our selection can intersect. Clearly, if we answer the group-canisters problem, we have found <= m sets whose union is U, so we have also answered the set-cover problem. Suppose on the other hand that we answer the set-cover problem and find k or less (possibly intersecting) sets that cover U. For each element that appears more than once, arbitrarily remove all copies of it but one. The resulting sets are a solution of the group-canisters problem. The problem is NP-complete.

4.) Galactic Shortest-Path Problem

The problem is in NP because we can always check a path against the limits on its costs by summing the costs of the edges in linear time.

Per a tip from Dr. R, I'll use subset sum. Create a residual graph in which all vertices appear twice (a mirror of the old graph), except s and t, which join the two graphs. The edges in the first graph have the costs from L and the edges in the mirrored graph have costs from R. Direct all the edges in the first graph from s to t and all the edges in the mirrored graph from t to s, and find a cycle from s to s. Find the set of paths with length less than or equal to half the sum of the edge costs in the two graphs. If and only if such a path exists, it is possible to split the graph into two equal sum sets, so the problem reduces to subset sum and is NP complete.