Homework 26

Brian Knotten, Brett Schreiber, Brian Falkenstein

October 28, 2018

22

23

Define TRI to be the triangle problem described in question 22, and DCP to be the Dr. Cuddy problem defined for this question. The goal is to show that $TRI \leq_{poly} DCP$.
The input to TRI is a set W composed of the cartesian product of 3 size n sets X, Y, Z. Note that not all of $X \times Y \times Z$ is in W, only a subset of all the possible cartesian products. Further, because every element of X, Y, Z must be in exactly one element of the output set U, and thus $|U| = n$, we will only consider instances of the problem where $|W| \geq n$, as otherwise there could not be a subset of $|W|$ of size n, and a solution would not exist.
Assume the sets X, Y, Z are readable, and thus a part of the input to TRI, as the membership of these sets is required to determine the validity of a solution, as each member of each set must appear exactly once in the solution.
For each set member $x_i \in X$, create a disease a_{xi} for input to DCP. Do this for all members of Y and Z as well. This results in $3n$ diseases being created for input to DCP. Note that presently, no tests exist and thus a solution cannot be found (as there is no way to distinguish each disease).
For each tuple $w_i \in W$ of the form (x_i, y_j, z_k) (where x_i denotes the i'th member of X, y_j denotes the j'th member of Y, etc), construct the following:

- 3 tests T_{xi}, T_{yj}, T_{zk} s.t. $T_{xi}(a_{xi}) = 1$, $T_{yj}(a_{yj}) = 1$, and $T_{zk}(a_{zk}) = 1$, and all 3 tests return 0 for any other input. That is, the disease a_{xi} created corresponding to $x_i \in X$ now has a test T_{xi} that can distinguish it from all other diseases, in that $T_{xi}(a_{xi}) = 1$ and $T_{xi} = 0$ for all other inputs

Further, we will assign $j = 3n$ (where n is the size of the sets X, Y, Z).
The claim is that a solution to the DCP instance exists if and only if one exists for TRI. We can show this both ways:
If we assume that the input to TRI does not have a solution, and thus there does not exists a subset U of W that is of size n where each element in X, Y, Z appears exactly once in U, we can show that the instance we've constructed for DCP also has no solution. Call the element that appears in either X, Y or Z but not in U (there may be more than one) x_i. In the first step of constructing our input, we made a disease a_{xi} corresponding to this element. If x_i does not appear in any $w \in W$, then its corresponding test T_{xi} was never constructed, and all other tests constructed $\forall T_k \in T$, $T_k(a_{xi}) = 0$. Thus, we'd have no solution to the DCP instance as no test exists that can distinguish between a_{xi} and any other disease.
If we assume the input to TRI does have a solution, we can show that a solution also exists in the DCP instance constructed. Because we constructed diseases for each member of X, Y, Z, and they may only be distinguished from other diseases by a test constructed when that member was found in the input W, it cannot be the case that some $x_i \in X$ exists that is not in the solution U, but DCP returns true. Further, because we assign $j = 3n$, we can accept only those solutions that distinguish all diseases (we constructed 3n diseases, n for each of X, Y, Z). Thus, if some subset $U \in W$ exists s.t. each member of X, Y, Z appears exactly once in U, then we will have a set of $3n$ tests that can distinguish between all a_{xi}, a_{yj}, and a_{zk}, as we will have taken those tests constructed for each member of U to be our solution to DCP.
This transformation of inputs takes poly time. The first step, constructing the diseases, will take $3n$ time, as we must create 1 disease (constant time) for each of n members in X, Y, Z, resulting in $3n$ total diseases being made. Then, 3 tests must be constructed (constant time to make a test. as its just a simple function) for each member of W. We've already noted that $|W| \geq n$, but we must also note that $|W| \leq n^3$, as the size of the cartesian product of 3 size n sets is n^3. So, we will construct somewhere between $3n$ and $3n^3$ tests for our instance of DCP. This results in a runtime of $O(3n + 3n^3) = O(n^3)$. Thus, we have proven that if we have a poly-time algorithm for DCP, then we will also have a poly-time algorithm for TRI, showing that DCP is NP-Hard.

24

We will reduce the Vertex Cover problem to the Fox, Goose, and Bag of beans puzzle, hereafter referred to as the FGB problem. In order to do so, we will create a graph H and integer ℓ that that represent a boat of size ℓ that can safely transport the objects represented by the vertices of H iff there exists a vertex cover of size k for the graph G. In order to construct H, copy all of the vertices and edges from G and add two additional vertices x and y that are disconnected from all other vertices except each other. ℓ is merely $k + 1$.
This transformation is clearly polynomial in the size of G, as H is constructed using the size of G + a constant number of vertices and edges.
To prove G has a vertex cover of size k iff there exists a boat of size ℓ that can safely transport the objects represented by the vertices of H, we must prove the relationship both ways:

First, assume there exists a boat of size ℓ that can safely transport the objects represented by the vertices of H. If so, then consider an arbitrary trip using the boat. Because there is an edge between x and y, then it must be the case that x and y are not both on the boat or both on one of the shores. The vertices remaining on each shore must be safe to be together i.e. must both be an independent set. However, if this is the first trip across, then one of x or y must be on the boat. Therefore there are at most $\ell - 1 = k$ of G's vertices on the boat and both the vertices on the boat and the vertices on the shore are an independent set. Thus, the $\leq k$ vertices on the boat during the first trip are a vertex cover of G.

Second, assume G has a vertex cover of size k. Then a boat of size $k + 1 = \ell$ can safely carry the items represented by H by putting the vertex cover of G and the new vertex x in the boat. Because a vertex cover touches every edge of a graph, the vertices left on the shore (including y) are an independent set of H and can therefore remain on the shore safely. We can then deposit x on the second shore. The vertices in the vertex cover of G must remain on the boat as the vertices on the first shore (i.e. those not in the vertex cover) will be transported across one at a time, so we cannot leave a vertex in the vertex cover on the second shore with one not in the vertex cover. We cannot simply deposit the vertex cover with x and then transport the remaining vertices in one trip, as the number of vertices not in the vertex cover + y may be greater than $k + 1$. Therefore we will deposit x first, then transport the vertices not in the vertex cover one by one, and then transport y last and all the vertices in H have been safely transported by a boat of size $\ell = k + 1$.

Therefore, given a poly-time algorithm that solves the FGB problem, we can solve Vertex Cover in polynomial time by using the poly-time transformation given above, a call to the poly FGB algorithm, and outputting 1 if the poly-time algorithm does, and 0 otherwise