Homework 23

Brian Knotten, Brett Schreiber, Brian Falkenstein

October 21, 2018

14

a

The idea is to reduce Clique to an instance of Clique with a fixed k, specifically where $k = 3n/4$. Call this new problem $3n/4Clique$. Obviously, if the k for Clique already equals $3n/4$, no transformation needs to be made, and the output of $3n/4Clique$ would equal the output of $Clique$.
In the case where k is greater than $3n/4$, we can construct G' as input to $3n/4Clique$ by adding vertices to G, such that all the added vertices have no edges connecting them to any other vertices, until $3n/4 = k$. These new vertices will obviously not contribute to any cliques existing in G, as they have no connections. Now, if $3n/4Clique$ returns true for G', we know that there must be a clique of size k in G, as G' maintains all the cliques from G, with some added disconnected vertices to ensure that $k = 3n/4$. IE it could not be the case that G' has a clique of size $3n/4$ while G does not have a clique of size k, as those $3n/4 - k$ vertices that were added could not contribute to any cliques. This transformation certainly takes polynomial time, as it simply requires adding a constant number of disconnected vertices.
In the case where k is less than $3n/4$, we can construct G' by adding cliques of size $(3n/4) - k$ to every vertex V, where each vertex in the new added clique connects to V, as well as all neighbor vertices to V in G. Now, any vertex in G that previously belonged to a clique of size i belongs to a clique of size $i + (3n/4) - k$. Thus, if a clique of size k exists in G, a clique of size $(3n/4) - k + k = 3n/4$ exists in G'. Further, if no clique of size k exists in G, there won't be a clique of size $3n/4$ in G', as none of the newly added cliques are connected to each other, and only add $(3n/4) - k$ to each already existing clique in G. This transformation will take polynomial time in the input, as we will need to add n new cliques to the graph, each of constant size, and need to make at most n new connections for each added clique (must connect to each vertices neighbors, vertices can have at most $n - 1$ neighbors).

b

c

This is an NP-hard problem, because a poly-time algorithm for this problem implies a poly-time algorithm for the clique problem.

CliqueAlgorithm(G, k):
    Let $G' = G + k$ new vertices not connected to any other vertices.
    return CliqueAndIndependentSetAlgorithm(G, k)

The fact that G' always has an independent set of k vertices implies that CliqueAndIndependentSetAlgorithm on G' will return true if and only if G' (and therefore G) has a clique of size k. This algorithm is poly-time because the transformation adds only a linear number of vertices.

d

e

This is an NP-hard problem, because a poly-time algorithm for this problem implies a poly-time algorithm for the 3n/4 clique problem.

3n/4CliqueAlgorithm(G):
    Let $G' = G + 3n/4$ new vertices not connected to any other vertices.

return 3n/4CliqueAndIndependentSetAlgorithm(G)

The fact that G' always has an independent set of $3n/4$ vertices implies that CliqueAndIndependentSetAlgorithm on G' will return true if and only if G' (and therefore G) has a clique of size $3n/4$. This algorithm is poly-time because the transformation adds only a linear number of vertices.

16

A 3-COLOR algorithm makes a choice at every vertex v_i either to color it red, green, or blue. We can model that in a Boolean formula as three variables r_i, g_i, b_i, such that only one of the three can be true. So in the formula F there is a clause $(r_i \oplus g_i \oplus b_i)$, which in CNF is:

$$\begin{aligned}&(\neg r_i \vee \neg g_i)\wedge\\&(\neg g_i \vee \neg b_i)\wedge\\&(\neg b_i \vee \neg r_i)\wedge\\&(r_i \vee g_i \vee b_i)\end{aligned}$$

Moreover, for each edge $(v_i, v_j) \in G$, it cannot be the case that they are the same color, which corresponds to the variables r_i and r_j not both being true, same with g_i and g_j, and b_i and b_j. So in the formula F there is a clause $(\neg(r_i \wedge r_j)) \wedge \neg(g_i \wedge g_j) \wedge \neg(b_i \wedge b_j))$, which in CNF is:

$$\begin{aligned}&(\neg r_i \vee \neg r_j)\wedge\\&(\neg g_i \vee \neg g_j)\wedge\\&(\neg b_i \vee \neg b_j)\end{aligned}$$

So an algorithm for 3-COLOR is:

3-COLORAlgorithm(G):
    Let $F = \emptyset$
    for each vertex $v_i \in G$:
        $F = F\cup$ (vertex CNF described above)

    for each edge $(v_i, v_j) \in G$:
        $F = F\cup$ (edge CNF described above)

    return CNF-SATAlgorithm(F)

It must be shown that F is satisfiable if and only if G is 3-Colorable.

If G is 3-Colorable, then each vertex is one of $\{r, g, b\}$. For each vertex v_i colored c, let c_i be true, and the other two colors associated with i to be false in F, and this will satisfy F.

If F is satisfiable, then for all i, one of r_i, g_i, b_i is true. Use this to color the vertex v_i and it will give a valid 3-Coloring of G.

Moreover, it must be shown that the transformation from G to F is polynomial. Since each vertex yields 4 clauses and each edge yields 3 clauses, which are both constant-size, the transformation is still polynomial in the number of vertices and number of edges, $O(n^2)$.

17

In order to reduce the Vertex Cover problem to the Dominating Set problem, given the graph G and desired cover size k, we will construct a graph G' such that G' has a dominating set of size k iff G has a vertex cover of size k. Let V and E be the set of all vertices and edges in G respectively. We will construct G' as follows:

V' contains all the vertices in V as well as new utility vertices corresponding to each edge in E.

$V' = \{v|v \in V\} \cup \{v'|(u,v) \in E\}$

E' contains all the edges in V as well as new edges connecting each utility vertex to the endpoints of the edge it was derived from. Moreover, all vertices originally in V have new edges between them.
$E' = \{(u,v),(u,v'),(v',v)|(u,v) \in E\} \cup \{(u,v)|(u,v) \in V \times V \wedge u \neq v\}$.

To prove that G' has a dominating set of size k iff G has a vertex cover of size k we must prove the relationship in both directions:
First, assume G has a vertex cover VC of size k. Then G' has a dominating set $DS = VC$ of size k as every vertex $v \in VC$ has a representative vertex u in G' that is connected to every other vertex in G' representing a vertex in G and every vertex in G' representing an edge in G must connected to at least one vertex in VC because at least one of that edge's endpoints in G must be a vertex in VC by the definition of a vertex cover.

Second, assume G' has a dominating set DS of size k. If DS contains any vertices representing an edge in G, replace it with one of the vertices representing an endpoint of its edge in G. DS is still a dominating set because any edge-representing vertex removed is next to its replacement (one of its endpoints) and every vertex that was next to the removed vertex is also next to its replacement (every endpoint is connected to every other endpoint by our construction). By definition, the vertices of G' are all either in DS or are connected by an edge to a vertex in DS. We can therefore construct a vertex cover VC of G by taking every vertex in G that is a vertex represented by a vertex in DS i.e. $VC = DS$. VC must be a vertex cover of G as for any edge in G there must be a vertex in DS that is connected to the vertex representing that edge in G' by the definition of a dominating set. Because $VC = DS$, that vertex is one of the endpoints of that edge in G and is in VC.