Assignment 2 - Problem 3.5, 4.4

Kousshik Raj (17CS30022)

20-09-2020

1 Problem 3.5

In the **CLUSTER EDITING** problem, we are given a graph G and an integer k, and the objective is to check whether we can turn G into a cluster graph (a disjoint union of cliques) by making at most k edge editions, where each edition is adding or deleting one edge. Obtain a $3^k n^{O(1)}$ - time algorithm for **CLUSTER EDITING**.

1.1 Solution

Let G be the given graph. Now we want to check, if we can turn G into a disjoint union of clusters with at most k editions. Let P_3 denote an induced path of length 3.

Claim 1. *G is a cluster graph if and only if there is no P_3 in G.*

Proof Let G be a cluster graph. Then the shortest distance between any two vertices belonging to the same connected component is 1, as each vertex is connected to every other vertex in its connected component. But for P_3 to exist, there should at least be two vertices where the shortest distance between them is 2. And as P_3 only consists of vertices belonging to the same connected component, we can say that G doesn't contain P_3.

For the other direction, let G be a graph which doesn't contain P_3. This means that there is no two vertices belonging to the same connected component where the shortest distance between them is greater than 1, as otherwise there will be a P_3 present. So, distance between every pair of vertices in the same connected component will be equal to 1, which further implies that every pair of vertices is connected by an edge if they belong to the same connected component. Thus, G is a cluster graph. $\square$

By *Claim 1*, as long as there is a P_3 in G, G cannot become a cluster graph. So the main idea of the following algorithm is to add or remove an edge for every P_3 we find in G, and we stop when we exceed our budget or when there is no P_3 in G.

Claim 2. **CLUSTER EDITING** problem can be solved in $O(3^k n^{O(1)})$ time.

Proof Let the given instance be (G, k). First, let's look at the following algorithm that solves the **CLUSTER EDITING** problem.

We first find a P_3, say A, in G. If there is no such A, then the graph is already a cluster graph and we return that (G, k) is a YES-instance. Else, if $k \leq 0$, we return that it is a NO-instance. Otherwise, let x, y, z be the three vertices in A such that (x, y), $(y, z) \in E(G)$ and $(x, z) \notin E(G)$. To remove this P_3 from G, we can either remove the edges (x, y), (y, z) from G or add the edge (x, z) to G. So, we branch on all the 3 options. In the first branch, we solve for the instance $(G - \{(x, y)\}, k - 1)$. Similarly, in the second branch, we do the same for the edge (y, z). In the third branch, we try to solve for the instance $(G \cup \{(x, z)\}, k - 1)$. If all the three branches, return that it is a NO-instance, we return that (G, k) is also a NO-instance. On the other hand, if one branch returns that it is a YES-instance, we add the corresponding edge to the solution X, and return that (G, k) is a YES-instance.

If we look at the recursion tree, we see that each level has at most 3 branches, and the total depth of the tree is at most k, as at each level k decreases by 1. This means the total number of nodes in the tree is of the order $O(3^k)$ and each node takes $O(n^{O(1)})$ time. Therefore, the running time of the algorithm is $O(3^k n^{O(1)})$. □

Thus we have a given an algorithm that can solve the **CLUSTER EDITING** problem in $O(3^k n^{O(1)})$ time.

2 Problem 4.4

An undirected graph is called *perfect* if for every induced subgraph H of G, the size of the largest clique in H is the same as the chromatic number of H. We consider the **ODD CYCLE TRANSVERSAL** problem, restricted to perfect graphs. Find a $O(2^k n^{O(1)})$ time algorithm based on iterative compression.

2.1 Solution

Let G be a given graph that is *perfect.*

Claim 1. The odd cycle transversal for the perfect graph G is a set of vertices that has non-empty intersection with every cycle of length 3 in G.

Proof Odd cycle transversal of a graph is a set of vertices such that it has non-empty intersection with every cycle of odd length. But since G is a perfect graph, according to *Strong Perfect Graph Theorem*, it does not have odd cycle of length greater than 3. Hence, if X has non-empty intersection with every cycle of length 3 in G, it is an odd cycle transversal of G. □

We now aim to find an algorithm based on iterative compression that finds a set of vertices with non empty intersection with all cycle of length 3. Take an arbitary ordering $(v_1, v_2, ..., v_n)$ of $V(G)$. Let G_i be the subgraph induced by taking only the first i vertices and X_i be the corresponding odd cycle transversal.

It can be seen that $X_i = V(G_i)$ is a odd cycle transversal of G_i of size at most k, $\forall i \leq k$. We now iteratively make our instance bigger until it is the same as the original instance G. Suppose that for some $i \geq k$, we have constructed an odd cycle transversal X_i of G_i with size at most k. Now, we look for an odd cycle transversal X_{i+1} of G_{i+1} of size at most k. Let $Z_{i+1} = X_i \cup \{v_{i+1}\}$. It can be seen that Z_{i+1} is an odd cycle transversal of G_{i+1}. If $|Z_{i+1}| \leq k$, we are done and we can proceed to the next iteration with $X_{i+1} = Z_{i+1}$. Otherwise, we try to compress Z_{i+1} to size at most k or determine that it is impossible. If we can solve this *compression* in $O(2^k n^{O(1)})$ time, we have an algorithm that solves odd cycle transversal for perfect graphs in $O(2^k n^{O(1)})$ as well.

Now, the compression algorithm takes the perfect graph G, its odd cycle transversal Z with $|Z| = k+1$, and k as input, and now we have to find an odd cycle transversal X such that $|X| \leq k$ or determine that there is no such X. We now try out all possible ways X intersects Z. Let $X \cap Z = X_z$. Since X_z is part of the solution we can now remove it from the original instance. Now in the modified instance we want to find an odd cycle transversal Y such that $|Y| \leq k - |X_z|$ and $Y \cap Z = \phi$. If there is a possible Y for at least one of these scenarios, then we can return the corresponding $X_z \cup Y$ as a solution and we can be done. But, if there is no such Y for any of the possible intersections, then there is no solution X of size at most k for the given instance. Moreover, if we solve this *disjoint* version in polynomial time, i.e, in $O(n^{O(1)})$ time, we can solve the *compression* in $O(2^k n^{O(1)})$.

All that is left is to solve the *disjoint* version in polynomial time. We are given a perfect graph G, its odd cycle transversal W, and an integer k, and now we have to find an odd cycle transversal X such that $|X| \leq k$ and $X \cap W = \phi$. Let us call a cycle of length 3 as a triangle. According to *Claim 1*, it is enough if we remove all triangles from G. Let $A = V(G) - W$. Here we should carry out some reductions.

- **RED 1** If $G[W]$ has any triangle, it is a NO-instance, as we cannot remove any vertices from W.
- **RED 2** If a vertex $v \in A$ is not a part of any triangle in $G[W \cup A]$, remove the vertex.
- **RED 2** If $G[W \cup A]$ has a triangle with exactly 1 vertex in A, then add that vertex to the solution, and decrease k by 1.

Note that $G[A]$ cannot have any odd cycles, because W is a odd cycle transversal of G. So, after all the reductions, we now have triangles in G such that exactly two of its vertices are in A. and all vertices in A are part of some triangle. So, if we remove any one of the vertex for all such cycles we will have a graph without any triangles. This is the same as finding the vertex cover for $G[A]$. If the found solution has size at most k, we can return the solution. Otherwise, we say that it is a NO-instance. Moreover, since $G[A]$ is bipartite (as there are no odd cycles), we can find the vertex cover, and thus equivalently solving the *disjoint* version of the problem, in polynomial time, i.e, in $O(n^{O(1)})$ time.

With this, as we solved the *disjoint* version in $O(n^{O(1)})$, which made the *compression* algorithm run in $O(2^k n^{O(1)})$, we solved the **ODD CYCLE TRANSVERSAL** problem, restricted to perfect graphs, in $O(2^k n^{O(1)})$ time.