COL351 Assignment 4

Tanish Tuteja, Harshit Mawandia

November 2022

Q1

1

Let $I = (U, A_1, A_2, ..., A_m$ be a YES instance of the problem. For it to be an NP class problem, the solution must be verifiable in polynomial time.
Let S be the solution. The verification algorithm is as follows:

Algorithm

$ans \leftarrow TRUE$
for all $A \in \{A_1, A_2, ..., A_m\}$ **do**
  if $A \cap S = \emptyset$ **then**
    $ans \leftarrow FALSE$
    break
  end if
end for
if $|S| > k$ **then**
  $ans \leftarrow FALSE$
end if

Proof of Correctness

The proof of correctness follows from the definition of Hitting Set itself.

Time Complexity

We iterate over all the sets A_i, which takes $O(m)$ time. In each iteration, we take intersection of A_i with S, which can be computed in $O(M_i log M_i)$ time, where $M_i = max(|A_i, |S|)$. Thus, overall time complexity,

$$T(m, n) = O(m)O(\max_i M_i)$$

$$= O(nm)$$

Thus, proved the problem is in NP class.

2

Consider an instance of vertex cover problem $G = (V, E)$. Now for each edge $e_i = (x, y) \in E$, make a corresponding set $A_i = \{x, y\}$. Also, let $U = V$. Then, hitting set problem (U, A_i) is a mapping of vertex cover problem $G = (V, E)$.

Proof

Let $S \subseteq V$ be a solution of the vertex cover problem.
$\Rightarrow \forall e = (x, y) \in E$ either $x \in S$ or $y \in S \Rightarrow S \cap \{x, y\} \neq \emptyset$
$\Rightarrow S \cap A_i \neq \emptyset \forall i$
Also, $|S| \leq k$ since it is a solution of vertex cover problem. Thus, it is also a solution of the Hitting Set problem.

Conversely, let $S \subseteq U = V$ be a solution of the hitting set problem generated.
$\Rightarrow S \cap A_i \neq \emptyset$
$\Rightarrow S \cap \{x, y\} \neq \emptyset \forall e = (x, y) \in E$
$\Rightarrow \forall e = (x, y) \in E$, either $x \in S$ or $y \in S$ Also, $|S| \leq k$ since it is a solution of hitting set problem. Thus, it is also a solution of the vertex cover problem.

Since Hitting Set problem is in NP class as proved earlier, and $HittingSet \geq VertexCover$, and Vertex Cover is NP-Complete, Hitting Set problem is also NP-Complete.

Q2

(i)

Let $I = (G = (V, E), s, t)$ be a YES instance of the problem. For it to be an NP class problem, the solution must be verifiable in polynomial time.
Let T be the solution. The verification algorithm is as follows:

Algorithm

```
function Paths-To(currPath, currNode, parents, ans)
    if currNode = NULL then
        toAdd ← currPath.reverse()
        ans.append(toAdd)
        return
    end if
    for all node ∈ parents[currNode] do
        currPath.append(currNode)
        Paths-To(currPath, node, parents, ans)
        currPath.removeLast()
    end for
end function

function Find-All-Paths(G = (V, E), s, t)  ▷ Finds all paths in G between s and t
    for all n ∈ V do
        dist[n] ← NULL
        parents[n] ← []
    end for
    bfsQ ← Queue( )
    bfsQ.enqueue(s)
    parents[s] = [-1]
    while bfsQ ≠ ∅ do
        n ← bfsQ.dequeue()
        for all u s.t. (n, u) ∈ E do
            if dist[u] = NULL then
                dist[u] ← dist[n] + 1
                parents[u].append(n)
                bfsQ.enqueue(u)
            else if dist[u] = dist[n] + 1 then
                parents[u].append(n)
            end if
        end for
    end while
    ans ← []
    Paths-To([], t, parents, ans)
```

```
    return ans
end function

function CHECK(G = (V, E), s, t, T)
    P ← FIND-ALL-PATHS(G, s, t)
    ans ← TRUE
    for all p_1 ∈ P do
        for all p_2 ∈ P, p_2 ≠ p_1 do
            if T ∩ p_1 = T ∩ p_2 then
                ans ← FALSE break
            end if
        end for
    end for
    if —T— ¿ k then
        ans ← FALSE
    end if
end function
```

(ii)

To prove that Tracking Shortest Path Problem is NP-complete, we try to reduce the problem from Vertex Cover problem.

Let $\{G(V,E),k\}$ be any instance of the vertex cover problem. We construct the graph $G'(V',E')$ from $G(V,E)$ as follows:

We create a vertex in G' for every vertex in G and also for each edge in G. We also create vertex s and t, so that $V' = V \cup E \cup \{s,t\}$. In G' there is an edge between a vertex $v \in V$ and a vertex $e \in E$ if e is incident on v in G. For each vertex $v \in V$, we create an edge between v and s in G'. and for each vertex $e \in E$ we create an edge between e and t in G'. Finally let $B = E$ be the given set of trackers

Now, lets consider any $tracking-set$ $A \subset V'$ for the set of all shortest paths between s and t. We can see that any shortest path between s and t is of length 3. Any vertex $e \in V'$ corresponding to an edge $e \in E$ is incident on two vertices v_i and v_j such that e is the edge between v_i and v_j in G. To distinguish between paths (s, v_i, e, t) and (S, v_j, e, t), A must contain either v_i and v_j. Hence $A \setminus B$ is vertex cover for G.

Now, let us consider any vertex cover $V* \subset V$ for G. We show that $V* \cup B$ is a $tracking - set$ for all shortest paths from s to t in G'. Suppose there exists two paths (s, v_i, e_j, t) and (s, v_k, e_l, t) which are indistinguishable. If $e_j \neq e_l$, then clearly (s, v_i, e_j, t) and (s, v_k, e_l, t) are distinguishable because $e_j, e_l \in B$. Hence without loss of generality assume $e_j = e_l$. Therefore, as (s, v_i, e_j, t) and (s, v_k, e_l, t) are indistinguishable, both v_i and v_k are not in $V*$. But this contradicts the fact that $V*$ is a vertex cover for G. Hence the result holds.

Claim: There exists a vertex cover for graph G of size k if and only if there exists a Tracking Set for G' of size $k + |E| + 1$.

Proof. Let $G(V, E)$ be any graph. We begin by constructing the graph $G'(V', E')$ from $G(V, E)$ as before. Next, we add to V' the vertices a, b, d and to E' the edges $(s, a), (s, b), (d, t)$. We also add to E' the edges (a, x), for each $x \in E \cup \{d\}$, and the edge (b, d).

Let $|V*| = k$ be any vertex cover for G. Consider the Tracking Set $V* \cup E \cup \{a\}$ for G'. Observe that it is possible to track all shortest paths between s and t passing through d, because there are only two such paths, one with a single tracker at a and the other without any tracker. It is also possible to track all paths of the form (s, a, e_k, t), because each of them has two trackers, one at a and one at e_k. Suppose now that there exist two paths (s, v_i, e_k, t) and (s, v_j, el, t) which are indistinguishable. This implies that $e_k = e_l$ and that both v_i and v_j are not present in $V*$. But this contradicts the fact that $V*$ is a vertex cover for G.

Now suppose there exists a Tracking Set T for G' of size $k + |E| + 1$. To distinguish between the paths (s, a, x, t), where $x \in E \cup \{d\}$, all vertices in $E \in \{d\}$ except one must contain a tracker. We may assume that each of the vertices in E contains a tracker (and d does not), since otherwise, we can simply move the tracker at d to e_i, where e_i is the sole vertex in E that does not contain a tracker. Clearly, the new set of trackers is still a Tracking Set for G' of size $k + |E| + 1$.

Observe that T contains either a or b to distinguish between the paths (s, a, d, t) and (s, b, d, t). As (s, b, d, t) is the only path passing through b and $E \subset T$ we may assume that there is a tracker at a (otherwise, we simply move the one at b to a). Now $T \setminus (E \cup \{a\}) \subset V$ is of size k, and it is easy to verify that $T \setminus (E \cup \{a\})$ is a vertex cover for G.

Claim TSPP is NP-complete.

Proof From class we know that vertex cover is NP-complete, which is the hardest problem in NP set. But we have just proved that vertex cover problem can be reduced to TSPP. This means that TSPP is at least as hard as vertex cover problem, which implies that TSPP is also NP-complete.

Q3

(i)

Let $E(A, B)$ represent the number of edges with one endpoint in A and other in B. Thus, we require a set $S \subseteq V$ such that

$$E(S, S)/|S| \geq \alpha$$

Consider a flow network $G^{'} = (V^{'}, E^{'})$ constructed as follows:

$$V^{'} = V \cup \{s, t\}$$

$$E^{'} = E \cup \{(s, i)|i \in V\} \cup \{(i, t)|i \in V\}$$

with capacities

$$c_{ij} = 1 \forall (i, j) \in E$$

$$c_{si} = |E| \forall i \in V$$

$$c_{it} = |E| + 2\alpha - deg(i) \forall i \in V$$

Let (S, T) be the s-t mincut for this network.

Claim: $S - \{s\} \neq \emptyset$ **iff** $\exists S \subseteq V$ **s.t.** $\frac{E(S,S)}{|S|} \geq \alpha$
Proof:
Let $S - \{s\} = V_1 \neq \emptyset$ and $V_2 = V \setminus V_1$
Capacity of this min cut

$$C(S, T) = \sum_{i \in V_2} c_{si} + \sum_{i \in V_1} c_{it} + \sum_{i \in V_1, j \in V_2} c_{ij}$$

$$= |E||V_2| + (|E||V_1| + 2\alpha|V_1| - \sum_{i \in V_1} deg(i)) + E(V_1, V_2)$$

$$= |E||V| + 2\alpha|V_1| - \sum_{i \in V_1} deg(i) + E(V_1, V_2)$$

Consider another partition of $G^{'}$, $(S^{'}, T^{'})$ s.t. $S^{'} = \{s\}$ and $T^{'} = V^{'} \setminus S^{'}$
Capacity of this cut $C(S^{'}, T^{'}) = |E||V|$
This must be greater than (or equal to) the min cut,

$$\Rightarrow |E||V| \geq |E||V| + 2\alpha|V_1| - \sum_{i \in V_1} deg(i)) + E(V_1, V_2)$$

$$\Rightarrow 2|V_1| \left(\alpha - \frac{\left(\sum_{i \in V_1} deg(i)\right) - E(V_1, V_2))}{2|V_1|} \right) \leq 0$$

$$\Rightarrow \alpha \leq \frac{\sum_{i \in V_1} deg(i)) - E(V_1, V_2)}{2|V_1|}$$

$$\Rightarrow \alpha \leq \frac{E(V_1, V_1)}{|V_1|}$$

Conversely, let $V_1 \subseteq V$ s.t. $\alpha \leq \frac{E(V_1,V_1)}{|V_1|}$
As calculated before, the capacity of this cut is $|E||V|+2\alpha|V_1|-\sum_{i\in V_1} deg(i)+E(V_1, V_2)$

If possible, let $S = \{s\}$ Thus, the capacity of the min-cut is $|E||V|$ This should be less than the capacity of V_1 cut.

$$\Rightarrow |E||V| < |E||V| + 2\alpha|V_1| - \sum_{i\in V_1} deg(i) + E(V_1, V_2)$$

$$\Rightarrow 2\alpha|V_1| > \sum_{i\in V_1} deg(i) + E(V_1, V_2)$$

$$\Rightarrow \alpha > \frac{\sum_{i\in V_1} deg(i) + E(V_1, V_2)}{2|V_1}$$

$$\Rightarrow \alpha > |E(V_1, V_1|$$

which is a contradiction Thus, $S \neq \{s\}$

Therefore, proved.

Algorithm

$G^{'}$ = Graph-To-Network(G) ▷ As defined earlier
$(S, T) \leftarrow$ Min-Cut($G^{'}, s, t$)
if $S = \{s\}$ **then**
  $ans \leftarrow FALSE$
else
  $ans \leftarrow TRUE$
end if

The proof follows from the preceding claim.

Time Complexity

The conversion of graph to network requires $O(n + m)$ time. The computation of min-cut using Ford-Fulkerson max flow algorithm takes $O(nm(n+m))$ time. Thus, overall time complexity,

$$T(n, m) = O(n + m) + O(nm(n + m)$$

$$= O(nm(n + m))$$

(ii)

We know the maximum value of $E(S)$ can be $|E| = m$, and the minimum value of $|S| = 1$. Thus, maximum density $= |E| = m$. Also, minimum density can be 0 when $E(S) = 0$ Thus, we can use perform a binary search in the range $[0, m]$ and use the algorithm from part (i) to find the maximum density and corresponding subset of vertices.

Algorithm

$start \leftarrow 0$
$end \leftarrow m$
$result \leftarrow \emptyset$ ▷ Stores the final answer
while $start \leq end$ **do**
  $middle \leftarrow \left\lfloor \frac{(start+end)}{2} \right\rfloor$
  $ans, S = \textsc{CheckGrEq}(G, middle)$ ▷ Ans is the min-cut as computed in part (i)
  if $ans = TRUE$ **then**
    $start \leftarrow middle + 1$
    $result \leftarrow S \setminus \{s\}$
  else
    $end \leftarrow middle - 1$
  end if
end while

The proof again follows from the claim made earlier.

Time Complexity

The binary search will run a max of $O(logm)$ iterations, and in each iteration the algorithm of part (i) is called once. Thus, overall time complexity

$$T(n, m) = O(logm)O(mn(m + n))$$

$$= O(nm(logm)(m + n))$$