CSC373 Homework 4

April 25, 2022

Question 1 Approximate Vertex Cover

(a) The possible values that any variable x_v can take in an optimal solution to the above ILP is 0, or 2. We use the value of x_u to represent whether u is in the vertex cover, therefore it is a binary value. We also need to satisfy both the requirements that $x_u + x_v \geq 2$ for each edge (u, v), and $x_u \geq 0$ for each vertex. Therefore, the possible values that any variable x_v can take in an optimal solution to the above ILP is 0, or 2.

(b) Prove: $\forall(x, y) \in E$, either x or y are not in S.
We know that the LP relaxation has an optimal solution x_v^* in which $x_v^* \in \{0, 1, 2\}$, sine we need to satisfy both the requirements that $x_u + x_v \geq 2$ for each edge (u, v), and $x_v \geq 0$ for each vertex.
Prove by contradiction.
Assume $\exists(x, y) \in E$ st both x and y are not in S. Then, by the definition of S, we know $x, y = 0$ or 1.
Case 1: $x = y = 0$, then the equation $x + y \geq 2$ is violated (contradiction).
Case 2: x, y are 0, and 1 respectively, then the equation $x + y \geq 2$ is violated (contradiction).
Case 3: $x = y = 1$, then since they both are not in S, they must have the same color 3, which contradicts to the 4-coloring assumption of the graph G.

(c) Let $V^m = \{v \in V, x_v^* = m$, where $m \in 0, 1, 2\}$, and $V_t^m = \{v \in V, c_v = t \ \& \ x_v^* = m$, where $m \in 0, 1, 2\}$.
The objective to minimize is $obj(X) = 0.5 * (|V^1| + 2 * |V^2|) = 0.5 * |V^1| + |V^2|$ (1).
From the question, without loss of generality, assume $|V_3^1| \geq |V_2^1| \geq |V_1^1| \geq |V_0^1|$.
Rewrite $|V_3^1| \geq |V_2^1| \geq |V_1^1| \geq |V_0^1|$ as:$|V_3^1| \geq |V_2^1|, |V_3^1| \geq |V_1^1|, |V_3^1| \geq |V_0^1|, |V_3^1| = |V_3^1|$
Sum them up (left side by left side; right side by right side), we can get $4|V_3^1| \geq |V^1|$, or equally $|V_3^1| \geq \frac{1}{4}|V^1|$.
Then

$$
\begin{aligned}
|V_3^1| &\geq \frac{1}{4}|V^1| \\
-|V_3^1| &\leq -\frac{1}{4}|V^1| \\
|V^1| - |V_3^1| &\leq \frac{3}{4}|V^1| \\
|V_1^1| + |V_2^1| + |V_3^1| &\leq \frac{3}{4}|V^1|, (2)
\end{aligned}
$$

From the definition of S, we know that $|S| = |V^2| + |V_0^1| + |V_1^1| + |V_2^1|$.

$$
\begin{aligned}
|S| &= |V^2| + |V_0^1| + |V_1^1| + |V_2^1| \\
&\leq |V^2| + \frac{3}{4} \cdot |V^1|, \text{ from (2)} \\
&\leq \frac{3}{2} \cdot obj(X) - \frac{1}{2}|V^2| \leq \frac{3}{2} obj(X)
\end{aligned}
$$

Since we are using LP relaxation that calculates the optimum vertex cover, we know that the optimum vertex cover x^* uses at least $obj(X)$ vertices. Therefore we can get $|S| \leq \frac{3}{2} obj(X) \leq \frac{3}{2} x^*$.

(d) Similar to the proof as in (c), we can prove the new equation (2) as $|V_1^1| + |V_2^1| + ... + |V_k^1| \leq \frac{k-1}{k}|V^1|$. The $obj(X)$ is the same. Then we can prove similarly that $|S| \leq 2(\frac{k-1}{k}) obj(X) \leq 2(\frac{k-1}{k}) x^*$. Therefore the worst case approximation ratio of this vertex cover for G is $2(\frac{k-1}{k})$.

Question 2 Nearly-SAT

(a) We can take an assignment S as the advice, and the algorithm is to check each clause while keeping track of the number of clauses that evaluate to TRUE. Checking each clause takes $\mathcal{O}(n)$ time and there are m clauses, so the algorithm is in $\mathcal{O}(mn)$, which is polynomial. This algorithm can accept every YES instance with the right polynomial-size advice, and will not accept a NO instance with any advice, so that nearly-SAT is in NP.

(b) Let p be the last clause in $\mathcal{F}$. Let $\mathcal{F}' = \mathcal{F} \wedge p \wedge (\neg p)$. For any assignment, only one of p and $\neg p$ will evaluate to TRUE, so an assignment that satisfies all m clauses of $\mathcal{F}$ satisfies precisely $m+1$ clauses of $\mathcal{F}'$.

(c) We know that SAT is NP-complete, and we will reduce SAT to Nearly-SAT to show that Nearly-SAT is NP-hard. We will show that $\mathcal{F}$ is satisfiable if and only if $\mathcal{F}'$ has an assignment satisfying $m+1$ clauses.

$\mathcal{F}$ is satisfiable $\Rightarrow$ $\mathcal{F}'$ has an assignment satisfying $m+1$ clauses: Take any assignment that satisfies $\mathcal{F}$. Since p is the same as the last clause in $\mathcal{F}$, it evaluates to TRUE, so $\neg p$ evaluates to FALSE, and $\mathcal{F}'$ has an assignment satisfying $m+1$ clauses.

$\mathcal{F}'$ has an assignment satisfying $m+1$ clauses $\Rightarrow$ $\mathcal{F}$ is satisfiable: We know that only one of p and $\neg p$ evaluates to TRUE. Since $\mathcal{F}'$ has an assignment satisfying $m+1$ clauses, and one of the clauses in $\mathcal{F}$ is the same as p, given this assignment, the only clause that evaluates to FALSE has to be $\neg p$. Removing p and $\neg p$ from $\mathcal{F}'$, $\mathcal{F}$ has exactly m satisfiable clauses.

Since both transforming $\mathcal{F}$ to $\mathcal{F}'$ and transforming $\mathcal{F}'$ to $\mathcal{F}$ takes polynomial time, it is a polynomial time reduction, and Nearly-SAT is NP-hard.

Question 3 Tile Covering

(a) Assume there are n tiles in the set. Let the advice be an arrangement of all tiles in the set, and the arrangement for the i^{th} tile is $\{i, x_i, y_i, \text{orientation}\}$, where i is the index of the tile, (x_i, y_i) is the bottom-left coordinate of the tile, and orientation is whether the tile is placed horizontally or vertically. The algorithm will include the following checks:

- Total area covered: To cover the walls with tiles, the total area of all tiles should be equal to the area of the wall, $\sum l_i w_i = l \times w$.
- No overlap: Iterate through the tiles and keep track of the outer boundary coordinates of the checked tiles. For each new tile, check that its coordinates do not fall inside the boundaries.
- All tiles are within boundary: Check that for each tile, $x_i \geq 0, y_i \geq 0$. For each horizontal tile, $x_i + l_i \leq l, y_i + w_i \leq w$; for each vertical tile, $x_i + w_i \leq l, y_i + l_i \leq l$.
- No uncovered holes: Check if the conditions of both total area is covered and no overlap is satisfied. If both are satisfied, then there is no hole on the wall.

Since the check for each of the four conditions runs in polynomial time, the algorithm is in polynomial time. This algorithm can accept every YES instance with the right polynomial-size advice, and will not accept a NO instance with any advice, so that Architect problem is in NP.

(b) We know that Partition is NP-complete, and we will reduce Partition problem to the Architect problem to show that the Architect problem is NP-complete.

Consider a very narrow rectangular region, let A be tiles of sizes $l_i \times \epsilon$, where ϵ is a very small value so we don't need to consider horizontal or vertical alignment. Let $s = \frac{1}{2}\sum_i l_i$, and let $B = s \times 2\epsilon$. There is a valid partition of 2 sets of $\{l_1, \cdots, l_n\}$ if and only if there is a valid arrangement of A into B. To prove the forward direction, we can place the first subset $\{l_1, \cdots, l_j\}$ to the upper row, and place the second subset $\{l_{j+1}, \cdots, l_n\}$ to the lower row. To prove the backward direction, we can place the lengths of tiles in the upper row to one subset, and place the lengths of tiles in the lower row to the other subset.

Since the transforming takes polynomial time, it is a polynomial time reduction, so Partition can be reduced to the Architect problem, and the Architect problem is NP-complete.