COMP 6651: Solutions to Assignment 10

Fall 2020

Submission through Moodle is due by November 29th at 23:55

1. Step 1: $SPECIAL-CLIQUE \in \mathcal{NP}$: the certificate is the set vertices S which forms a clique in G of size at least $|V|/2$. The certificate can clearly be checked in polynomial time.

   Step 2: We will show $SPECIAL-CLIQUE$ is $\mathcal{NP}$-hard by reducing from $CLIQUE = \{\langle G, k\rangle : G$ has a clique of size at least $k\}$.

   Step 3: let $G = (V, E), k$ be the input to $CLIQUE$. We need to construct an instance G' to $SPECIAL-CLIQUE$ such that

   $$\langle G, k\rangle \in CLIQUE \iff \langle G'\rangle \in SPECIAL-CLIQUE.$$

   Let $n = |V|$. Construction has two cases. Case $k < n/2$: we add $n-2k$ new vertices V_{new} to G. Each new vertex is connected by an edge to every vertex in G as well as to every other new vertex. The resulting graph is $G' = (V', E')$. Observe that if S is a clique in G of size at least k then $S \cup V_{\text{new}}$ is a clique of size at least $k + n - 2k = n - k$ in G'. Moreover, G' has $n + (n - 2k) = 2n - 2k$ vertices in total. Thus, $S \cup V_{\text{new}}$ is a clique of size at least $|V'|/2$ in G', as desired. In the opposite direction, suppose that S' is a clique in G' of size at least $|V'|/2 = n - k$. Then removing vertices V_{new} from S' gives a clique in G. This removes at most $n - 2k$ vertices, leaving a clique of size at least k in G.

   Case $k \geq n/2$. In this case, we simply add $2k-n$ new isolated vertices V_{new} to G and call the resulting graph $G' = (V', E')$. Observe that $|V'| = n + (2k - n) = 2k$. Moreover, a clique of size $|V'|/2$ cannot contain isolated vertices V_{new} (assuming $k > 1$ since otherwise problem is easy to solve). Thus, a clique of size at least $|V'|/2$ in G' is a clique of size at least k in G and vice versa.

   In either case, construction can be carried out in polynomial time.

2. $L_{\text{SCHEDULE}} = \{\langle F_1, \ldots, F_k, S_1, \ldots, S_\ell, h\rangle \mid \exists$valid schedule with $\leq h$ time slots$\}$

   Claim: L_{SCHEDULE} is $\mathcal{NP}$-complete.

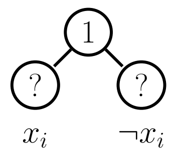
   Proof: L_{SCHEDULE} is in $\mathcal{NP}$: the certificate is the schedule mapping each exam to a time slot in $\{1, \ldots, h\}$. Note this certificate is of size polynomial in the size of the input, and it can be verified in polynomial time: we need to check that no two exams corresponding to each student are scheduled at the same time.

   We show how to reduce L_{COL} to L_{SCHEDULE} in polynomial time. Given an instance $\langle G, k\rangle$ we create an instance of L_{SCHEDULE} as follows. Let $G = (V, E)$ where $V = \{1, \ldots, n\}$ and $m = |E|$. We create n exams $F_1, \ldots, F_n$ and m. Thus, each exam corresponds to a node in G and each student i corresponds to an edge $\{u_i, v_i\} \in E$. Each student has exactly two exams. Let student i have exams F_{u_i} and F_{v_i}, i.e., $S_i = \{F_{u_i}, F_{v_i}\}$. We claim that $\langle F_1, \ldots, F_n, S_1, \ldots, S_m, k\rangle$ is in L_{SCHEDULE} if and only if $\langle G, k\rangle$ is in L_{COL}. Note that schedule mapping $\{F_1, \ldots, F_n\}$ to $\{1, \ldots, k\}$ can be interpreted as a colouring of nodes $\{1, \ldots, n\}$ with colors $\{1, \ldots, k\}$, and vice versa. The condition that no student has a conflict is equivalent to the condition that colouring is valid, i.e., no edge is coloured with the same color. Clearly, constructing the instance $\langle F_1, \ldots, F_n, S_1, \ldots, S_m, k\rangle$ takes polynomial time.

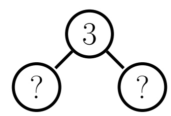
3. (a) $L_{\text{MINE}} = \{\langle G, \ell \rangle \mid \exists$ a placement of mines consistent with labels $\ell\}$.

(b) **Claim:** $L_{3SAT} \leq_p L_{\text{MINE}}$

Proof: Given a 3-CNF ϕ with n variables and m clauses, we show how to construct a labelled graph (instance of L_{MINE}) in polynomial time such that there is a placement of mines consistent with the labels if and only if the original formula ϕ is satisfiable. For each variable x_i introduce a variable gadget – three nodes such that two are unlabelled (we think of them as literals x_i and $\neg x_i$) connected to the third node labelled with 1. Thus, we can think of this gadget as follows: labelled node forces exactly one of its neighbours to contain a mine. The mine will be placed on the node x_i in case $x_i = 1$ in the satisfying assignment, and it will be placed on the node $\neg x_i$ if $x_i = 0$ in the satisfying assignment.

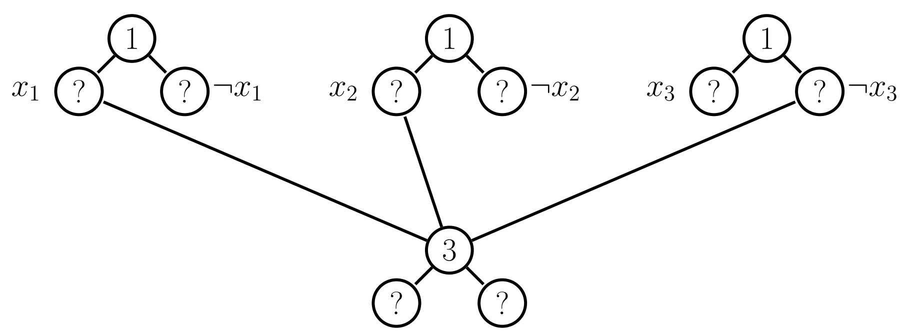


For each clause C_j we introduce a similar gadget, but the top node is labelled with 3 now:



Clause C_j

We connect the top node of the gadget corresponding to C_j to the bottom nodes of the corresponding literals in the variable gadgets. For example, if $C_j = x_1 \vee x_2 \vee \neg x_3$ we get the following subgraph:



$C_j = x_1 \vee x_2 \vee \neg x_3$

Clearly, this construction can be done in polynomial time. It is left to show that this instance has consistent placement of mines if and only if ϕ is satisfiable. Suppose that ϕ is satisfiable by assignment x. If $x_i = 0$ place a mine in the variable gadget i at the node corresponding to $\neg x_i$, otherwise place a mine at the node corresponding to x_i. Clause C_j can be satisfied by 1,2, or 3 literals. If clause C_j is satisfied by 3 literals, then placement of mines is already consistent with the clause gadget. If C_j is satisfied by 2 literals, then we place one more mine at one of the bottom nodes of the clause gadget to satisfy the label of the top node in the clause gadget. Lastly, if C_j is satisfied by 1 literal, then we place two mines at the unlabelled nodes of the clause gadget. This way we constructed a placement of mines consistent with all labels. It is now easy to see the other direction. Suppose that there is a placement of mines consistent with the labels. Then we can define as assignment x by setting x_i to 1 if and only if node corresponding to x_i has a mine in the variable gadget. This assignment satisfies every clause C_j, because the top node of the gadget corresponding to this clause has to have a neighbouring mine from one of the variable clauses, i.e., assignment x satisfies clause C_j.

(c) It is left to show that L_{MINE} is in $\mathcal{NP}$. The certificate is the placement of mines. Note that this certificate is polynomial in the size of the input, and it can be verified in polynomial time – for each labelled node we simply need to count the number of mines placed in the adjacent nodes and check that the label matches this number.