CS 721: Advanced Algorithms & Analysis

NP-complete reductions

1. **Prove that INDEPENDENCE-SET is NP-complete**

   There are two decision problems.

   3SAT: Given a CNF formula involving n clauses, where each clause contains at most 3 literals, is there a satisfying truth assignment to the literals so that the formula is true?

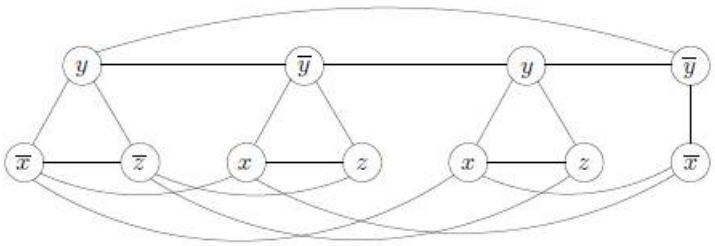
   INDEPENDENCE-SET: Given an input graph $G = (V, E)$ and an integer g, is there a set of g vertices so that no two vertices from this set are connected by an edge?

   Our goal is to prove that INDEPENDENCE-SET is NP-complete. In order to do that, we must show that (i) INDEPENDENCE-SET is in NP and (ii) a known NP-complete problem can reduced to INDEPENDENCE-SET.

   First we need to show that INDEPENDENCE-SET is in NP. Given an instance of INDEPENDENCE-SET problem and a candidate solution V' where $V' \subset V$ and $|V'| = g$, it is easy to check in polynomial time (in $|E|$ and $|V)$ whether no edges exists between any pair of vertices from V' or not.

   Next we will show how to reduce 3SAT to INDEPENDENCE-SET. Given an instance I of 3SAT, we create an instance (G, g) of INDEPENDENCE-SET as follows. Graph g has a triangle for each clause, that is each literal of a clause is a node of the graph and there is an edge between any pair of literals of a clause (or just an edge if there are only two literals).Create additional edges between any two vertices that represent opposie literals. Set goal g to be number of clauses. Clearly this construction can be done in polynomial time in the number of clauses. For the following instance of 3 SAT problem, we show the this construction explicitly.

   Figure 8.8 The graph corresponding to $(\overline{x} \vee y \vee \overline{z})\ (x \vee \overline{y} \vee z)\ (x \vee y \vee z)\ (\overline{x} \vee \overline{y})$.

   Next we need to show that, (i) If G has am independent set S of g vertices, how to efficiently recover a satisfying truth assignment for I, and (ii) If G has no independent set of size g then I has no satisfying truth assignment.

   - If G has an independent set S of g vertices, how to efficiently recover a satisfying truth assignment for I.

     For any variable x, the set S can not have both x and $\bar{x}$, because they will be connected by an edge. So if $x \in S$, assign true value to x, if $\bar{x} \in S$, assign false to x (if S contains neither, assign any value to x). Since S has g vertices, it must have one vertices per clause (and they can not be from the same clause because vertices from the same clause are connected to each other). Such truth assignment will ensure each clause is true and thus the CNF formula is true.

- If G has no independent set of size g then I has no satisfying truth assignment.

  It is easier to prove contrapositive, that is, if I has a satisfying truth assignment then G has an independent set of size g. This is easy. For each clause pick any literal whose satisfying assignment is true and add it to S. Then S must be of size g.

2. **Prove that VERTEX-COVER is NP-complete**

   There are two decision problems.

   INDEPENDENCE-SET: Given an input graph $G = (V, E)$ and an integer g, is there a set of g vertices so that no two vertices from this set are connected by an edge?

   VERTEX-COVER: Given an input graph $G = (V, E)$ and an integer b, is there a set of $V' \subset V$ of b vertices such that for any $(u, v) \in E$, either $u \in V'$ or $v \in V'$?

   Our goal is to prove that VERTEX-COVER is NP-complete. In order to do that, we must show that (i) VERTEX-COVER is in NP and (ii) a known NP-complete problem can reduced to VERTEX-COVER.

   First we need to show that VERTEX-COVER is in NP. Given an instance of VERTEX-COVER problem and a candidate solution V' where $V' \subset V$ and $|V'| = b$, it is easy to check in polynomial time (in $|E|$ and $|V)$ whether or not for every edge $(u, v) \in E$ either $u \in V'$ or $v \in V'$.

   Next we will show how to reduce INDEPENDENCE-SET to VERTEX-COVER. Observe that if a set of nodes S is a vertex cover for G, that is, one of the nodes of every edge of G is a member of S, then $V \setminus S$ is an independent set of G (can you see why?)

   Let (G, g) be an instance of INDEPENDENCE-SET. Then $(G, |V| - g)$ is an instance of VERTEX-COVER. This construct can be done in polynomial (in fact O(1)) time.

   Next we need to show that, (i) If G has a vertex cover of size $|V| - g$, how to efficiently recover an independence set of size g of G, and (ii) If G has no vertex cover of size $|V| - g$ then G has no independent set of size g.

   - If G has a vertex cover of size $|V| - g$, how to efficiently recover an independence set of size g of G,

     Simply choose the set of vertices that are not member of the vertex cover set.

   - If G has no vertex cover of size $|V| - g$ then G has no independent set of size g.

     It is easier to prove contrapositive, that is, if G has an independent set of size g then G has a vertex cover of size $|V| - g$. This is again easy. Suppose S of size g is an independent set of G. Then it is immediate that for every edge (u, v) of G, either $u \in V \setminus S$ or $v \in V \setminus S$.

3. **Prove that CLIQUE is NP-complete**

   There are two decision problems.

   CLIQUE: Given an input graph $G = (V, E)$ and an integer k, is there a set of $V' \subset V$ of k vertices such that all pairwise edges between V' is present in E?

   INDEPENDENCE-SET: Given an input graph $G = (V, E)$ and an integer g, is there a set of g vertices so that no two vertices from this set are connected by an edge?

   Our goal is to prove that CLIQUE is NP-complete. In order to do that, we must show that (i) CLIQUE is in NP and (ii) a known NP-complete problem can reduced to CLIQUE.

   First we need to show that CLIQUE is in NP. Given an instance of CLIQUE problem and a candidate solution V' where $V' \subset V$ and $|V'| = k$, it is easy to check in polynomial time (in $|E|$ and $|V)$ whether every pairwise edges in V' are present in E or not.

   Next we will show how to reduce INDEPENDENCE-SET to CLIQUE.

Let (G, g) be an instance of INDEPENDENCE-SET. Construct a new graph $G' = (V, E')$, where E' contains precisely those edges that are not present in G. Then a set of nodes S is an independent set of G if and only of S is a clique of G'. Therefoe (G', g) is an instance of CLIQUE. Clearly this construction can be done in polynomial time in $|V|$ and $|E|$.

Next we need to show that, (i) If G' has a clique of size g, how to efficiently recover an independence set of size g of G, and (ii) If G' has no clique of size g then G has no independent set of size g.

- If G' has a clique of size g, how to efficiently recover an independence set of size g of G
  Simply choose the set of vertices that are member of clique of G'
- If G' has no clique of size g then G has no independent set of size g.
  It is easier to prove contrapositive, that is, if G has an independent set of size g then G' has a clique of size g. This is again easy. Let S be an independent set of size g in G. By our construction all pairwise edges between vertices in S are added in G'. Therefore, S is a clique of size g in G'.

4. **Prove that TSP is NP-complete**

   There are two decision problems.

   TSP: Given an input graph $G = (V, E)$ a cost function $c : V \times V \to \mathbb{R}$ and a scalar $k \in \mathbb{R}$, is there a tour (i.e., visiting every vertex once and coming back to the starting vertex) of cost at most k?

   HAMILTONIAN-CYCLE: Given an input graph $G = (V, E)$ is there a simple cycle that contains each vertex in V?

   Our goal is to prove that TSP is NP-complete. In order to do that, we must show that (i) TSP is in NP and (ii) a known NP-complete problem can reduced to TSP.

   First we need to show that TSP is in NP. Given an instance (G, c, k) of TSP problem and a candidate solution, it is easy to compute the sum of the total cost of the tour in the candidate solution and check if it is less than k in polynomial time (in $|E|$ and $|V|$).

   Next we will show how to reduce HAMILTONIAN-CYCLE to CLIQUE.

   Let $G = (V, E)$ be an instance of HAMILTONIAN CYCLE. Construct a new graph $G' = (V, E')$, where G' is a complete graph and the edge weights (costs) are assigned as follows. If $(u, v) \in E$ then $c(u, v) = 1$ and if $(u, v) \notin E$ then assign $c(u, v) = 1 + \alpha$ for some $\alpha > 1$. Set $k = |V|$ then (G', c, k) is an instance of TSP. Clearly this construction can be done in polynomial time in $|V|$ and $|E|$.

   Next we need to show that, (i) If G' has a tour of cost at most $|V|$, how to efficiently recover a Hamiltonian cycle of G, and (ii) If G' has no tour of cost at most $|V|$ then G has no simple path that covers each vertex of V.

   - If G' has a tour of cost at most $|V|$, how to efficiently recover a Hamiltonian cycle of G
     Since a tour must visit each vertex of V exactly once, except the starting vertex, the tour must consists of $|V|$ edges and thus edge weight of each edge in the tour must be 1 (it can not be less than 1 by our construction). Note that by our construction, these are the edges which are also present in G. Therefore G contains a Hamiltonian cycle.
   - If G' has no tour of cost at most $|V|$ then G has no simple path that covers each vertex of V.
     It is easier to prove contrapositive, that is, if G has a Hamiltonian cycle the G' has a tour of cost at most $|V|$. Note that the Hamiltonian cycle of G is also a tour of G'. By our construction, the cost of this tour is $|V|$ since we assign edge weight (cost) 1 for each edge present in G.

5. **Prove that 3COLOR is NP-complete**

There are two decision problems.

3COLOR: Given an input undirected graph $G = (V, E)$ is it possible to color the graph (using three colors) using the function $c : V \to \{1, 2, 3\}$,such that $c(u) \neq c(v)$ for every edge $(u, v) \in E$?

3SAT: Given a CNF formula involving n variables and m clauses, where each clause contains exactly 3 literals, is there a satisfying truth assignment to the literals so that the formula is true?

Our goal is to prove that 3COLOR is NP-complete. In order to do that, we must show that (i) 3COLOR is in NP and (ii) a known NP-complete problem can reduced to 3COLOR.

First we need to show that 3COLOR is in NP. Given an instance G of 3COLOR problem, , and a candidate solution (in the form of a coloring function c), it is easy to check if $c(u) \neq c(v)$ for every $(u, v) \in E$ in polynomial time (in $|E|$ and $|V)$.

Next we will show how to reduce 3SAT to 3COLOR.

Let ϕ be an instance of a 3SAT formula involving n variables and m clauses. We next show how to construct an instance of a graph G_ϕ such that G_ϕ is 3-clolorable if and only if ϕ is satisfiable. The construction is as follows. We will use three colors T, F, and B. Each variable will be assigned a color. Color assignment T will imply the corresponding variable is set to be true. Color assignment F will imply the corresponding variable is set to be false. Color assignment B does not have any specific meaning.

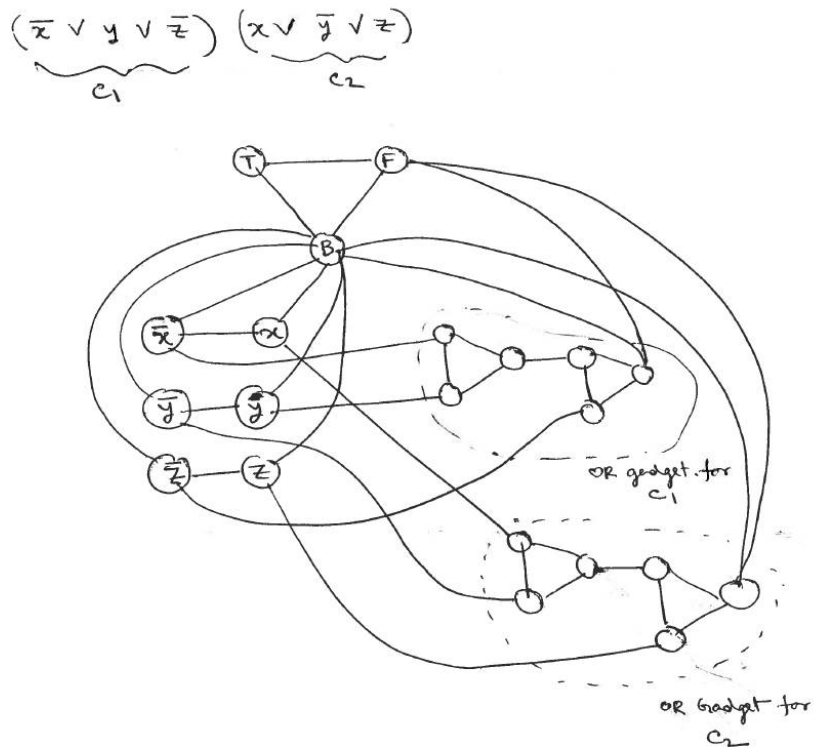
- Create a triangle with three nodes T, F and B. Clearly this is 3-colorable.
- For each variable x_i create two nodes x_i and $\bar{x}_i$. Connect these two nodes by an edge. Also form an edge between x_i and B and between $\bar{x}_i$ and B to form a triangle. This ensures that whenever the graph is 3-colorable x_i and $\bar{x}_i$ takes opposite colors.
- For each clause $(x_i \vee y_i \vee z_i)$ construct an OR gadget using two triangles such that this gadget takes input x_i, y_i, z_i and its output node is T if either x_i, y_i or z_i is T and output node is F if all x_i, y_i and z_i are F. Connect the output node of the OR gadget to both F and B. Clearly this construction can be done in time polynomial in m and n. For the following instance of 3 SAT problem, we show the this construction explicitly.

Next we need to show that, (i) If G_ϕ is 3-colorable then how to efficiently find a satisfying truth assignment for the variables of ϕ, and (ii) If G_ϕ is not 3-colorable then there is no satisfying truth assignment for ϕ.

- If G_ϕ is 3-colorable then how to efficiently find a satisfying truth assignment for the variables of ϕ

  If x_i is colored T then set x_i to true. This is a legal truth assignment. Consider any clause $(x_i \vee y_i \vee z_i)$. It is not possible that all are colored F because in that case output node of the OR gadget will be F which is connected to F, - meaning the graph is not 3 colorable. Thus, each clause has at least one variable which is assigned color T, ensuring that the clause is true.
- If G_ϕ is not 3-colorable then there is no satisfying truth assignment for ϕ.

  It is easier to prove contrapositive, that is, if there is a satisfying truth assignment for ϕ then G_ϕ is 3-colorable. This is easy to check. If x_i is assigned true, set color of x_i to T and $\bar{x}_i$ to F. For each clause $x_i \vee y_i \vee z_i$), at least one of x_i, y_i, z_i is true. This forces OR-gadget for clause to be 3-colorable such that its output node takes value T.


$(\bar{x} \vee y \vee \bar{z})$ $(x \vee \bar{y} \vee z)$
c_1
c_2
T
F
B
$\bar{x}$
x
$\bar{y}$
y
$\bar{z}$
z
OR gadget for c_1
OR Gadget for c_2