Demo

Presenter: Zhao Huangjie
Faculty of Engineering
Department of ISEM

October 20, 2020

Overview

Introduction

- **NP** is one of the most classical complexity classes that is widely studied by the theoretical computer science community.
- Many combinatorial problems such has Traveling Salesman Problem, Knapsack Problem, Packing Problem are proved to be in this class.
- A special subset of **NP** is **NP**-*complete* which contains the group of problems such that, if one of them can be solved with polynomial time deterministic algorithm, then all **NP** problem can be solved in polynomial time which further leads to $\mathbf{NP} = \mathbf{P}$.
- In this demo, we would like to show that *INDSET* problem is **NP**-*complete*.

Definition

- **NP**: A problem is in **NP** if it can be solved with non-deterministic Turing Machine in polynomial time.
- **P**: A problem is in **P** if it can be solved by a deterministic Turing Machine in polynomial time.
- **NP**-*hard*: A problem is in **NP**-*hard* if all problem in **NP** can reduce to it.
- Problem reduction: A problem A reduces to B in polynomial, denoted as $A \leq_p B$, means there exists a function f such that $x \in A \iff f(x) \in B$ and $f(x)$ can be computed in polynomial time.
- **NP**-*complete*: A problem is in **NP**-*complete* if it is in **NP** $\cap$ **NP**-*hard*.

Definition - *INDSET*

- *INDSET*: Given a graph $G = (V, E)$ with node set V and edge set E and an integer k, $(G, k) \in$ *INDSET* iff there exists an independent set $S \subseteq V$ in G with size $|S| \geq k$. An independent set S of G is defined as a subset of nodes of G with no edge $e \in E$ connecting any two of them. In other words, $(S \times S) \cap E = \emptyset$.

Definition - *INDSET*

- In following figure, the orange node can form an independent set.

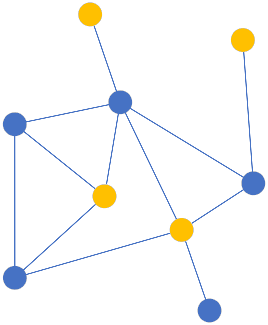

Figure: An example of independent set

Definition - *INDSET*

- In following figure, the red node is added, and there is an edge between the red node and one of the orange nodes, so that the nodes do not form an independent set.

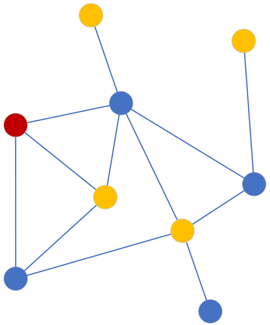

Figure: A set of node that is not independent

Definition - *3-SAT*

- *3-SAT*: A boolean formula ϕ in *3-CNF*(Conjective Normal Form) form of variable $u_1, u_2, \ldots, u_n$. Specifically, $\phi = \bigwedge_{i \in I} \left(\bigvee_{j \in s_i} v_j \right)$ where $|s_i| = 3$ for each i and $v_j = u_l$ or $\neg u_l$ for some l. Then $\phi \in$*3-SAT* iff $\exists u, \phi(u) = True$ which in other words, ϕ is satisfiable.
- An example: $\phi(u) = (u_1 \vee u_5 \vee \neg u_2) \wedge (u_2 \vee u_4 \vee u_6) \wedge (u_1 \vee u_3 \vee \neg u_2)$.
- It has been proved that $\mathit{3} - \mathit{SAT} \in \mathbf{NP} - \mathit{complete}$

INDSET is **NP**-*complete*

- To show that *INDSET* is **NP**-*complete*, we have to show that:
  - *INDSET* $\in$ **NP**
  - *INDSET* $\in$ **NP**-*hard*
- It is easy to observe that *INDSET* $\in$ **NP** since we can non-deterministically guess the independent set S, and check whether S is an independent set in polynomial time.
- To show that it is in **NP**-*hard*, we show that *3-SAT* $\leq_p$ *INDSET*. Since *3-SAT* is **NP**-*complete*, we can conclude that *INDSET* is **NP**-*hard* since for any problem L $\in$ **NP**, we have $L \leq_p$ *3-SAT* $\leq_p$ *INDSET*.

$3\text{-}SAT \leq_p INDSET$

- To show that *3-SAT* $\leq_p$ *INDSET*, we have to construct any *3-CNF* ϕ to an instance for *INDSET*. The construction is shown as follow:
- For each clause, we have a clique of 7 nodes represent 7 possible assignment of the three variable such that the clause is true. An example is shown as follow. Clause A is $u_2 \vee \neg u_{17} \vee u_{26}$.

Clause A

U2	0	0	0	1	1	1	1
U17	0	0	1	0	0	1	1
U26	0	1	1	0	1	0	1

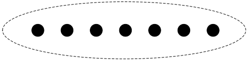

Figure: 7 nodes of a clause

3-SAT $\leq_p$ *INDSET*

- Then the a *3-CNF* formula with m clauses will have $7m$ nodes.
- In addition to the clique edges for each 7 nodes, we add edge between two small cluster if two nodes are conflict in assignment.
- Then we look for an independent set S that has size $|S| = m$.

3-SAT $\leq_p$ *INDSET* - example 1

- Clause A: $u_2 \vee \neg u_{17} \vee u_{26}$. Clause B: $\neg u_2 \vee \neg u_5 \vee u_7$, Clause C: $u_5 \vee \neg u_{27} \vee u_{43}$. Remark that only some of the edges are shown.

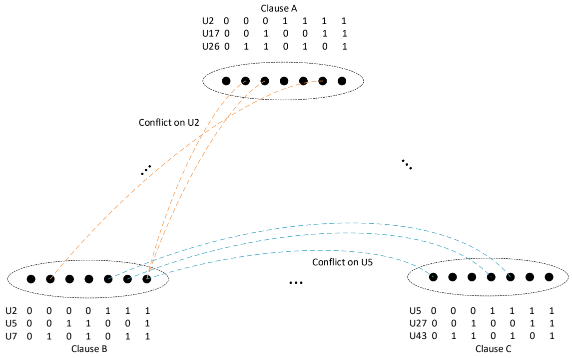

Figure: Construction of graph G

3-SAT $\leq_p$ *INDSET* - example 2

- Consider the case $\phi(u) = (u_1 \vee u_2 \vee \neg u_5) \wedge (u_2 \vee \neg u_3 \vee u_4) \wedge (\neg u_1 \vee \neg u_2 \vee u_3)$.
- Remark that only some of the edges are shown.

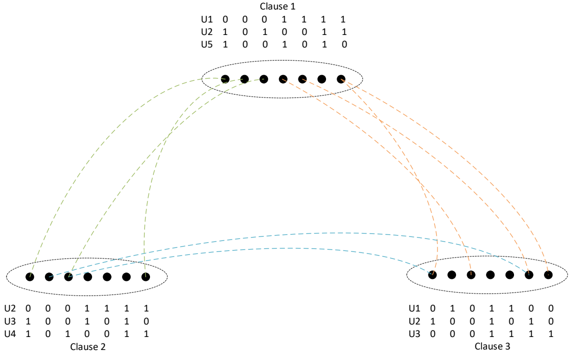

Figure: Construction of graph G for ϕ

3-SAT $\leq_p$ *INDSET*

- Then we need to show that each true assignment of variable u corresponds to an independent set of size m.
- Let's denote the edges set with in each small cluster as E_1 and the edges between two different cluster as E_2
- Given a formula ϕ, if it is satisfiable, then for each of the m small clusters, we pick the node that correspond to the assignment under u which is unique. Then it is easy to observe that this set is independent since all node follow same assignment u and therefore no edges from E_2 connects two nodes, and obviously no edges from E_1 connect two nodes as well.

3-SAT $\leq_p$ *INDSET*

- Given an independent set in the constructed graph with size m, we can firstly observe that in each cluster there is exactly one node that is chosen, otherwise an edge from E_1 will connect two nodes.
- Secondly we can observe that each variable must have exactly one assignment. We can look up the cluster that have the variable, and assign the variable according to the node that is chosen in the independent set. Since it is an independent set, the value for each variable is unique, otherwise an edge from E_2 will connect two nodes in the independent set which result in a contradiction.

3-SAT $\leq_p$ *INDSET*

- We can denote the construction discussed above as $f(\phi) =< G, m >$, and we therefore shows that $\phi \in$*3-SAT* $\iff f(\phi) \in$*INDSET*.
- Then we can conclude that *INDSET* is **NP**-*hard* and therefore **NP**-*complete*

Thank You!