# Lecture VII - NP Problems and Polynomial Transformation

[1] Department of Mathematics and Systems Analysis,
Systems Analysis Laboratory, Aalto University, Finland

February 29, 2024

Aalto University

Aalto University



# Previously on..

# PREVIOUSLY ON...

- Clique problems;
- Cover problems;
- Hamiltonian problems;

Also, all these problems can be easily "transformed" into each other. But **how**?

# Pack a knapsack!

What about integer linear inequalities, (decision version of) knapsack etc. ?

| | HP | Hunger Games | LotR | PJ&O | ATTWN | maximal weight |
|---|---|---|---|---|---|---|
| weight | 426g | 332g | 841g | 852g | 113g | 1000g |
| value | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | - |

- **input:** items $i$ with value $c_i$ and weight $w_i$
- **decision:** which items are packed
- **goal:** maximize value
- **constraints:** adhere to maximal weight $B$

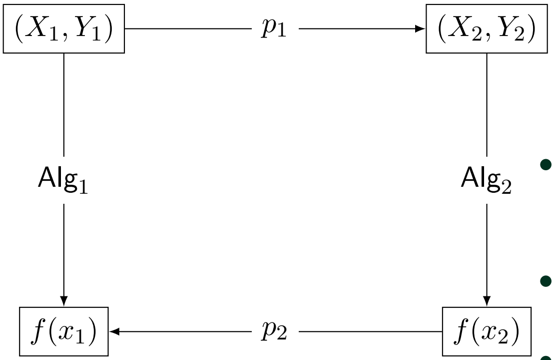
$$\max \sum_{i=1}^{n} c_i \cdot x_i$$

$$\text{s.t.} \sum_{i=1}^{n} w_i \cdot x_i \leq B$$

$$x_i \in \{0,1\}, \quad i = 1\ldots,n$$

# NP-Completeness

# Polynomial transformation




- $(X_1, Y_1)$ **polynomially transforms** to $(X_2 Y_2)$ if there exists polynomial function $p_1 \colon X_1 \to X_2$ such that

  $p_1(x_1) \in Y_2$ for all $x_1 \in Y_1$ and

  $p_1(x_1) \in X_2 \setminus Y_2$ for all $x_1 \in X_1 \setminus Y_1$

- yes-instances are mapped to yes-instances, no-instances are mapped to no-instances
- $(X_1, Y_1)$ is at most as hard as $(X_2, Y_2)$
- for general polynomial function $p_2$: **polynomial reduction**

Aalto University

# NP-completeness





## NP-completeness

$(X, Y) \in NP$ is called NP-complete if all other problems in NP **polynomially** transform to $(X, Y)$.

- NP-complete problems are the "hardest" problems in NP;
- if one NP-complete problem is solvable in polynomial time, all are $(P = NP)$;
- Do NP-complete problems exist?

# Satisfiability Problem (SAT)



literal: a binary variable, e.g. $x$, or its negation, e.g. $\neg x$

clause: a disjunction of literals, e.g.

$$x_1 \vee \neg x_2$$

CNF: conjunctive normal form, a conjunction of disjunction, e.g.

$$(x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge \neg x_4$$

SAT: satisfiability problem: Can a boolean formula, given as CNF, be satisfied?

# Cook's Theorem





Theorem (Cook, 1971)

*SAT is NP-complete.*

**Proof idea:**

- show that for any **nondeterministic algoritm** an equivalent SAT instance can be constructed in polynomial time

Need "narrow" definition of algorithms;

- many **thousand** problems have since been shown to be NP-complete
- **Karp's original 21 NP-complete problems**: Karp, R.M. (1975), *On the complexity of combinatorial problems.* Networks 5 (1975), 45–68

# Integer linear programming

Aalto University

## Theorem

*Integer linear programming is NP-complete.*

## Proof.

- idea: $(X, Y) \rightsquigarrow$ SAT $\rightsquigarrow$ integer linear programming
- check given solution in **polynomial** time $\Rightarrow$ integer linear programming is in NP
- let $F$ be a formula in CNF, **construct** ILP $P$
  - for each variable $x_i$ of $F$ **construct** a binary variable $y_i$ for $P$
  - for each clause $C$ **introduce one constraint** to $P$: $\sum_{i:\ x_i \in C} y_i + \sum_{i:\ \neg x_i \in C} (1 - y_i) \geq 1$
- $P$ is feasible $\iff$ $F$ is satisfiable

□

**Remark**: A problem that can be formulated as integer linear program is not automatically NP-complete.

# Knapsack problem

Aalto University

## Partition

**Instance:** $S$ multiset of positive integers.
**Question:** Can you partition $S$ into $S_1$, $S_2$ such that

$$\sum_{s \in S_1} s = \sum_{s \in S_2} s?$$

- known to be **NP-complete**

## Decision version of knapsack problem

**Instance:** Items $I$ with weight $w_i$, value $c_i$, $i \in I$, maximum weight $B$, minimum value $C$.
**Question:** Can you find $I' \subset I$ with

$$\sum_{i \in I} w_i \leq B$$

$$\sum_{i \in I} c_i \geq C?$$

- How can you **show** that knapsack is NP-complete?

# NP-hardness

- Problem $\mathcal{P}$ is called *NP-hard* if all problems in NP **polynomially** reduce to $\mathcal{P}$.
- $\mathcal{P}$ not necessarily in NP

**Examples**

- (optimization version of) knapsack
- multi-commodity flows
- traveling salesperson problem
- uncapacitated facility location

# NP, NP-Complete and NP-Hard

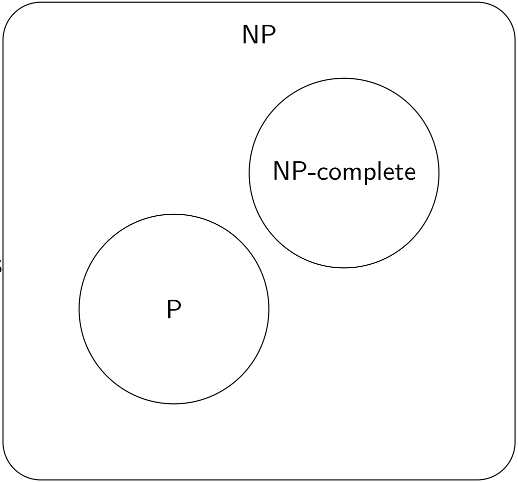


The difference between all three:

- NP: It is the collection of decision problems that can be solved by a non-deterministic machine in polynomial time.
- NP-Hard: An NP-hard problem is at least as hard as the hardest problem in NP and it is a class of problems such that every problem in NP reduces to NP-hard.
- NP-Complete: A problem is NP-complete if it is both NP and NP-hard. NP-complete problems are the hard problems in NP.

# $P \neq NP$?

- showing whether $P = NP$ or $P \neq NP$ is one of the **Millennium Prize Problems**
- most scientist believe that $P \neq NP$
- $P = NP$ would have large influences on the **(cyber) security** of **cryptography**




Aalto University

# $P \neq NP?$

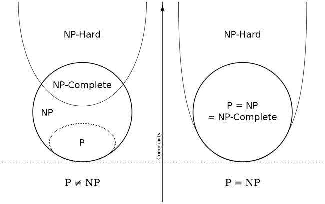



Figure: Question of the millennium?

# Polynomial transformation

# Recipe to prove that a problem is NP





- Show it is in NP:
  Verify that if a candidate solution is valid in polynomial time;
- Show it is NP-Hard:
  Reduce to a known NP-Complete problem.

With these two steps, **a novel problem can be considered a NP-Complete problem.**

# Clique and Independent Set

# Definition

Knowing that both **clique** and **independent set** are NP-Complete, there is a simple transformation between them:

**Clique and Independent Set Reduction:**

- For a graph $G = (V, E)$, build a complimentary graph $G'$;
- For every $v \in V$, it creates another set of nodes $v \in V'$;
- Add an edge in $G'$ for every edge not in $G$.

**Remark:** Complimentary graph can be calculated in polynomial time.

# Building Example

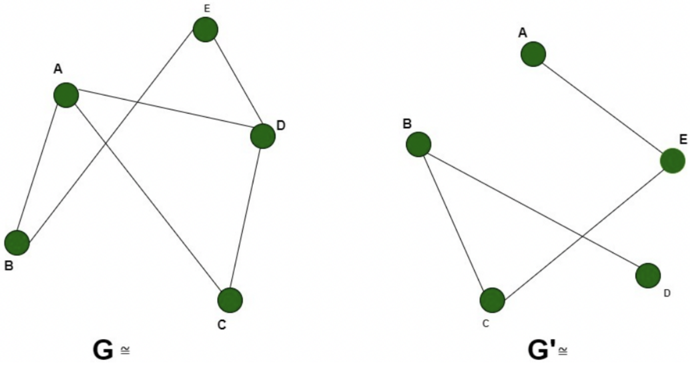



Figure: $G$ and complimentary $G'$

Combinatorial Optimization

Previously on..
NP-Completeness
Polynomial transformation
Clique and Independent Set
TSP and Hamiltonian Cycle
Independent Set and Vertex Cover
3-SAT to Clique

# Polynomial Reduction



*If* **there is** *an independent set of size* $k$ *in the complement graph* $G'$, **no two nodes share** *an edge in* $G'$. *Hence,* **all of those edges share an edge** *in* $G$ *forming a* **clique of size** $k$.

*If* **there is** *a clique of size* $k$ *in the graph* $G$, *all nodes share an edge in* $G$ *implying that there is* **no two nodes share** *an edge in* $G'$. *Hence,* **all of those edges share an edge** *in* $G'$ *forming an* **independent set of size** $k$.

# TSP and Hamiltonian Cycle

# Definition

Both problem are related to find a **cycle**.

**TSP and Hamiltonian Cycle reduction:**

- For a graph $G = (V, E)$, build a complimentary graph $G'$;
- For every pair of nodes $(u, v)$ without an edge in $G$, add an edge in $G'$.
- If edge $(u, v)$ exist in $G$, set the weight to zero, otherwise assign weight equal to one.

# Building Example

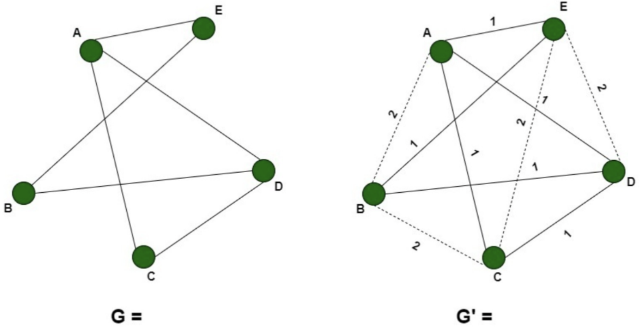



Figure: $G$ and complimentary $G'$

# Polynomial Reduction

The graph $G$ has a Hamiltonian cycle if **there is** a cycle in $G'$ passing through **all nodes only once with combined weight equal to zero**.

If the cycle passes through all nodes and the combined weight is zero, it means that the cycle **only contains edges present** in $G$. Hence, a **Hamiltonian cycle exists** in $G$.

If there is a Hamiltonian cycle in $G$, it also forms a **cycle** in $G'$ with combined weight equal to zero. Hence, a **solution for TSP** exists in $G'$.

# Independent Set and Vertex Cover

# Definition

Both problems can be traced to **covering** problems.

*If a graph $G$ has an* **independent set** *$S$, it also has a* **vertex cover** *$V - S$.*

# Polynomial Reduction



If $S$ is an independent set, there is no edge $(u, v) \in G$, such that both $v$ and $u$ are in $S$. Therefore, either $v$ or $u$ **has to be in** $V - S$.

If $V - S$ is a vertex cover, between any pair of nodes $u, v \in S$, the edge connecting them **would not exist** in $V - S$, otherwise it violates the definition of such vertex cover. Hence, no pair in $S$ can be reached by a single edge, creating an independent set.

**Remark:** Independent Set of size $k$ corresponds to a Vertex Cover of size $V - |k|$.

# 3-SAT to Clique

# Definition





A 3-SAT is composed from three-literal clauses. The goal is to reduce a clique of size $k$ in a group of $k$ clauses $\phi$.

- Building a graph $G$ of $k$ clusters with a **maximum** of 3 nodes in each cluster;
- Each cluster corresponds to a **clause** in $\phi$;
- Each node in a cluster is **labeled with a literal from the clause**;
- An edge is put between all pairs of nodes in different cluster **except for pairs of the form** $(x, \bar{x})$;
- **No edge is put between any pair of nodes in the same cluster**.

# Building Example

Given the following clause:

$$\phi = (x_2 + x_1 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + x_4)(x_2 + \bar{x}_4 + x_3)$$

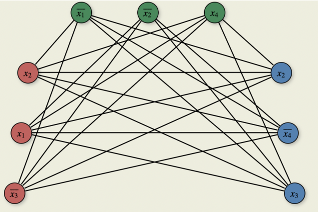



Figure: 3-SAT to clique

# Building Example

If **two nodes are connected**, it means that the literal can be simultaneously *true*.

If **two literals**, not in the same clause can be assigned *true* simultaneously; hence, the nodes are also connected.

# Polynomial Reduction

Aalto University

Combinatorial Optimization



$G$ has $k$-size clique, if $\phi$ is satisfiable.

If $G$ has a clique of size $k$, the clique has **exactly one node** in from each cluster. Hence, all corresponding literals can be assigned *true* with each literal belong to an **individual** $k$ **clauses**. Then, $\phi$ is **satisfiable**.

If $\phi$ is **satisfiable**, there is a combination of nodes corresponding to it. Let the set of nodes be $A$. From each clause, there are some literals that are *true*, that there are also in $A$. Remembering that **two literals cannot be from the same clause**, a clique can be formed by connecting a single node from each clause forming a **clique**.


thank you