Design and Analysis of Algorithms

Mohammad GANJTABESH

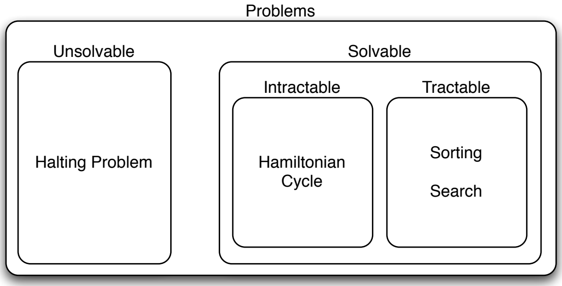
mgtabesh@ut.ac.ir

School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

Theory of *NP*—Completeness

Classification of the Problems

- Unsolvable: There is no algorithm to solve them.
- Intractable: Solvable in theory, but not solvable in practice, because of the huge amounts of required time.
- Tractable: Solvable both in theory and in practice.



Here we will focus on Intractable problems.

Deterministic vs. Nondeterministic

- Deterministic Algorithm: The next state of the algorithm execution can be definitely determined by the current state.
- Nondeterministic Algorithm: The next state of the algorithm execution can not be determined by the current state.

Based on the above principles, two classes of problems are as follows:

- P: The class of problems, for which a deterministic Polynomial-time algorithm exists.
- NP: The class of problems, for which a Nondeterministic Polynomial-time algorithm exists.

Question: How we can express a Nondeterministic Algorithm?

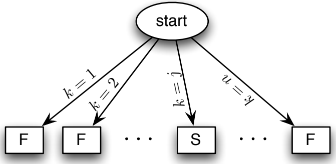
Express a Nondeterministic Algorithm

In order to express a nondeterministic algorithm, three functions, all with $O(1)$ time complexity are introduced as follows:

- choose(S): Concurrently create $|S|$ copies of the machine and assign one element of S to each copy.
- success(): Halt the machine with finding a solution.
- failure(): Halt the machine without finding a solution.

Example: Nondeterministic Search with $O(1)$ time complexity!

NSearch $(A[1 \cdots n], x)$
$k \leftarrow$ *choose*$(\{1, 2, \cdots, n\})$;
if $A[k] = x$ **then**
    success() ;
else
    failure() ;
end if
end.



Express a Nondeterministic Algorithm

Example: Nondeterministic Sort with $O(n)$ time complexity!

```
NSearch (A[1···n])
Array B[1···n] ← {0} ;
for i ← 1 to n do
    j ← choose({1, 2, ··· , n}) ;
    if B[j] ≠ 0 then
        failure() ;
    end if
    B[j] ← A[i] ;
end for
for i ← 1 to n − 1 do
    if B[i] ≠ B[i + 1] then
        failure() ;
    end if
end for
print(B) ;
success() ;
end.
```

A problem belongs to *NP* if one of the following holds.

- There is a nondeterministic polynomial-time algorithm for it.
- Verifying (deterministically and polynomialy) weather a given solution to that problem is correct or not .

Optimization vs. Decision

- Different optimization problems may have different outputs.
- In order to compare these problems, we need to have the same output for them.
- Any optimization problem can be converted to the corresponding decision problem by adding a bounding value.
- The answer of any decision problem is either yes or no.

Example (Knapsack Problem)

Suppose that we have n objects, say o_i ($i = 1, 2, \cdots, n$), each with corresponding weight (w_i) and profit (p_i), and a weight bound b.

- Optimization: Find an $X = (x_1, x_2, \cdots, x_n)$ that maximize $\sum_{i=1}^{n} x_i p_i$ with respect to $\sum_{i=1}^{n} x_i w_i \leq b$.
- Decision: For a given k, is there a feasible solution, say $X = (x_1, x_2, \cdots, x_n)$, where $\sum_{i=1}^{n} x_i p_i \geq k$?

Optimization vs. Decision

Example (Max-Clique Problem)

Suppose that a graph $G = (V, E)$ is given.

- Optimization: Find a maximal subset $V' \subseteq V$, such that the induced graph by V' is complete graph (The clique is a complete subgraph).
- Decision: For a given k, is there a subset $V' \subseteq V$ with $|V'| \geq k$, such that the induced graph by V' is complete graph (a clique of size at least k)?

Example (Min-Vertex Cover Problem)

Suppose that a graph $G = (V, E)$ is given.

- Optimization: Find a minimal subset $V' \subseteq V$, such that for each $e = (v_1, v_2) \in E$, either $v_1 \in V'$ or $v_2 \in V'$ (V' covers the E).
- Decision: For a given k, is there a subset $V' \subseteq V$ with $|V'| \leq k$, such that V' covers the E?

Optimization vs. Decision

Relation between Optimization and Decision problems

- If the optimization problem is easy to solve, then the corresponding decision problem is also easy.
- Conversely, If the decision problem is hard to solve, then the corresponding optimization problem is also hard.

Corollary

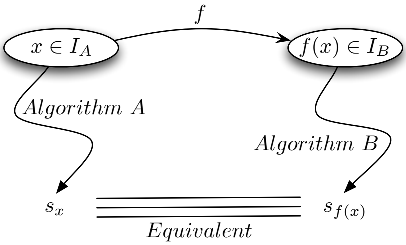
- *In order to show that the optimization problem is hard to solve, it is enough to show that the corresponding decision problem is hard!*
- *Dealing with decision problems is much easier than the optimization problems.*

The principle of Reduction

Definition

Suppose that A and B are two decision problems. We say that A is reduced to B (denoted by $A \preccurlyeq_P B$), if there exists a polynomial algorithm, say f, such that:

- $x \in \mathit{Instance}(A) \Longrightarrow f(x) \in \mathit{Instance}(B)$.
- x is a yes-instance of $A \Longleftrightarrow f(x)$ is a yes-instance of B.



Application of reduction

The reduction defines an order over the decision problems with respect to their level of difficulties.

- If B is easy to solve, then A is also easy.
- Conversely, If A is hard to solve, then B is also hard.

The theory of $NP-$Completness

Definition

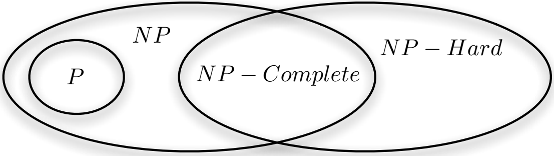
A problem L is called $NP-$Hard if all NP problems are reduced to L, i.e.

$$L \in NP - Hard \iff \forall L' \in NP \; : \; L' \preccurlyeq_P L.$$

Definition

A problem L is called $NP-$Complete if all NP problems are reduced to L and $L \in NP$, i.e.

$$L \in NP - Complete \iff L \in NP \cap NP - Hard.$$



The theory of $NP-$Completness

- Question: How we can prove that a problem is $NP-$Complete?
- Answer: Use the definition (very hard).

Theorem

The reduction has the transitive property, i.e.

$$A \preccurlyeq_P B \ \& \ B \preccurlyeq_P C \Longrightarrow A \preccurlyeq_P C.$$

- Question: How we can prove that a problem is $NP-$Complete?
- Answer: Use the above theorem (relatively easy). But we need at least one known $NP-$Complete problem.

Theorem (Cook Theorem)

The Satisfiability problem is an $NP-$Complete problem.

The theory of $NP-$Completness

Definition (Satisfiability problem)

Suppose that a formula ϕ over the n binary variables is given as follows:

$$\phi = \bigwedge_{i=1}^{m} C_i, \quad C_i = \bigvee_{j=1}^{k_i} \ell_{ij}, \quad \ell_{ij} \in \{x_{ij}, \overline{x}_{ij}\}, \quad k_i \in \mathbb{N}^+.$$

Is there an assignment $X = [x_1, x_2, \cdots, x_n] \in \{0,1\}^n$ such that ϕ is satisfied by the assignment X?

Definition (kSAT)

A SAT is called kSAT if $k_i = k$, for all $i = 1, 2, \cdots, m$.

Theorem

3SAT is an NP−Complete problem.

Theorem

2SAT is a polynomial-time solvable problem.

The theory of $NP-$Completness

Theorem

$k-$Clique is an $NP-$Complete problem.

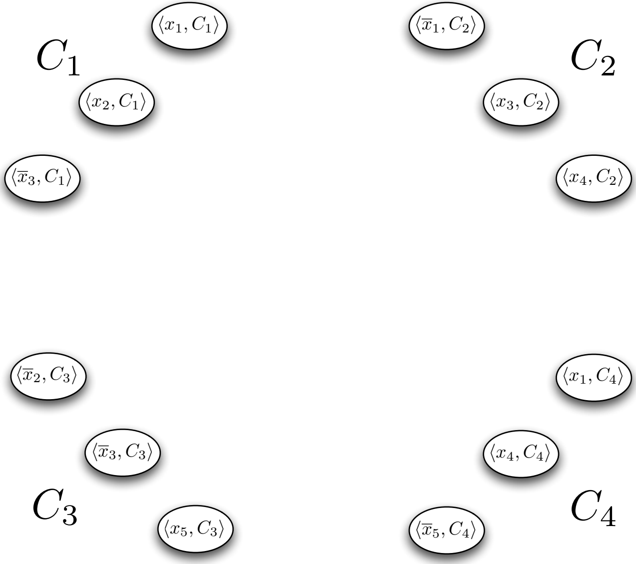
Example

Consider the following $3SAT$ formula:

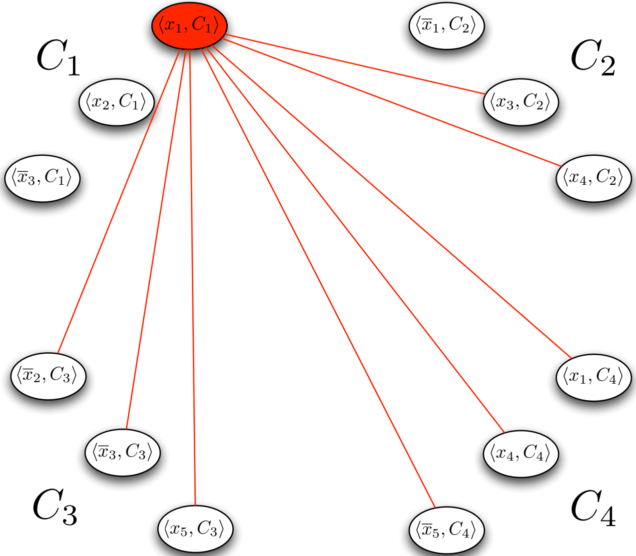
$$\phi = (x_1 \vee x_1 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_3 \vee x_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_5) \wedge (x_1 \vee x_4 \vee \overline{x}_5).$$

We can construct the corresponding graph $G(\phi)$ as follows (see the next slides).

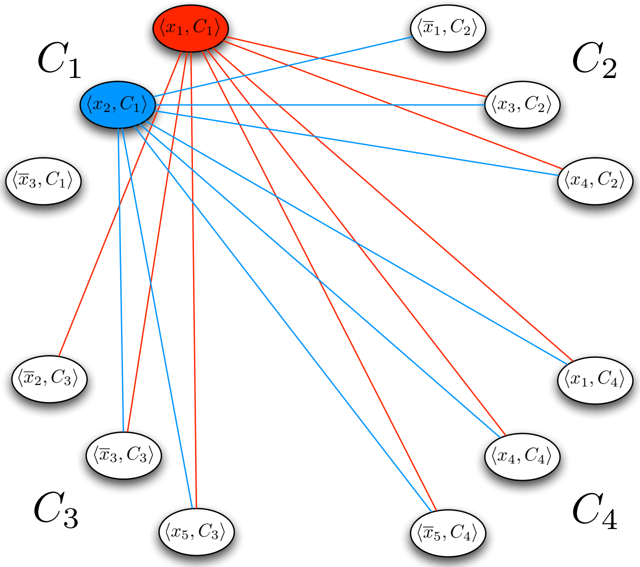
The theory of *NP*—Completness



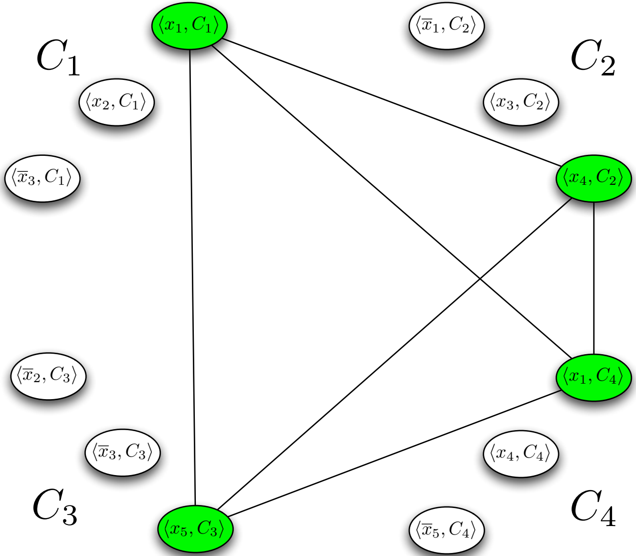
The theory of *NP*—Completness



The theory of *NP*—Completness



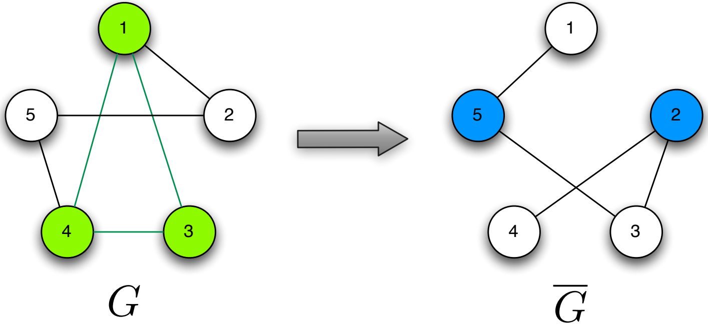
The theory of NP−Completness



The theory of *NP*−Completness

Theorem

k−Vertex Cover is an NP−Complete problem.



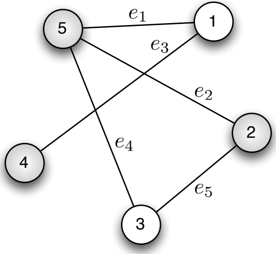
The theory of *NP*−Completness

Definition (Subset Sum Problem)

Suppose that a set $S = \{x_1, x_2, \cdots, x_n\}$ and a positive integer t are given. Is there a subset $S' \subseteq S$ such that $\sum_{x \in S'} x = t$?

Theorem

Subset Sum Problem is an NP−Complete problem.



M=

	e_1	e_2	e_3	e_4	e_5
v_1	1	0	1	0	0
v_2	0	1	0	0	1
v_3	0	0	0	1	1
v_4	0	0	1	0	0
v_5	1	1	0	1	0

The theory of *NP*—Completeness

	MSB	e_1	e_2	e_3	e_4	e_5	مبنای ۱۰
x_1	1	1	0	1	0	0	1296
x_2	1	0	1	0	0	1	1089
x_3	1	0	0	0	1	1	1029
x_4	1	0	0	1	0	0	1040
x_5	1	1	1	0	1	0	1348
y_1	0	1	0	0	0	0	256
y_2	0	0	1	0	0	0	64
y_3	0	0	0	1	0	0	16
y_4	0	0	0	0	1	0	4
y_5	0	0	0	0	0	1	1
t	k	2	2	2	2	2	3754

The theory of $NP-$Completeness

Definition (Hamiltonian Path (Cycle))

For a given graph $G = (V, E)$, a path (cycle) p is called Hamiltonian path (cycle) if it passes through all vertices and visit each vertex exactly once.

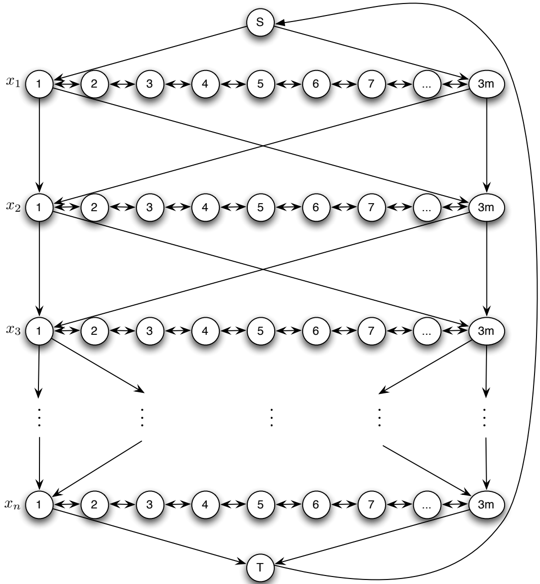
Definition (Hamiltonian Cycle Problem)

For a given graph $G = (V, E)$, check weather G has a Hamiltonian cycle or not?

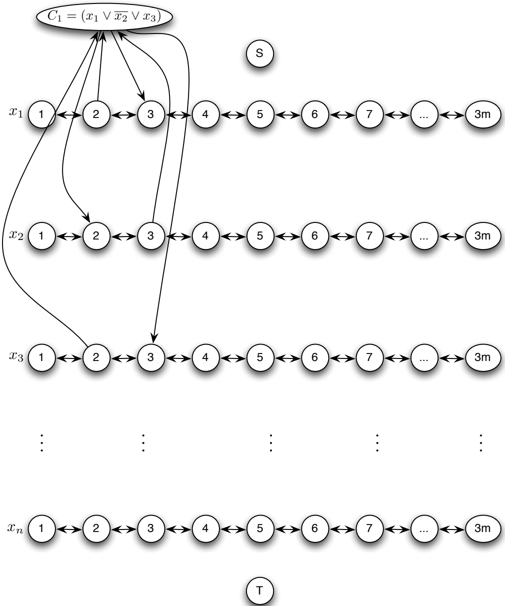
Theorem

Directed Hamiltonian Cycle (DHC) is an $NP-$Complete problem.

The theory of *NP*−Completness



The theory of *NP*−Completeness



The theory of $NP-$Completness

Definition (3Coloring Problem)

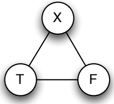
For a given graph $G = (V, E)$, check weather the vertices of G can be colored by three different colors in such a way that for all $e = (v_1, v_2) \in E$, $Color(v_1) \neq Color(v_2)$?

Theorem

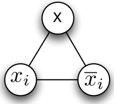
3Coloring is an $NP-$Complete problem.

The theory of *NP*−Completness

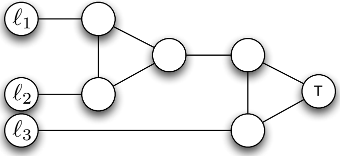
- Truth structure:



- Variable Structure:



- Clause Structure:



The theory of NP−Completness

$$\phi = (a \vee b \vee c) \wedge (b \vee \overline{c} \vee \overline{d}) \wedge (\overline{a} \vee c \vee d) \wedge (a \vee \overline{b} \vee \overline{d}).$$