NP-completeness

Selected from **Computer and Intractability** by M.R. Garey and D.S. Johnson (1979)

Provable intractability

- Two causes of intractability:
  1. The problem is so difficult that an exponential amount of time is needed to discover a solution.
  2. The solution itself has a length beyond any polynomial function of the input size.
     - E.g., given a weighted graph G and an integer B, list all Hamiltonian circuits having length of B or less.
     - Intractability of this sort can be easily recognized.
     - This type of intractability can be regarded as a signal that the problem is not defined realistically, because we are asking for more information than we could ever use.

Decision problems

- Only two possible answers: "yes" or "no".
- Problem description, two parts:
  1. Problem name.
  2. *Generic instance* of the problem.
  3. *Yes-no question* asked in terms of the generic instance.

- **PRIME**
- **Instance**: An integer $a \in Z^+$.
- **Question**: Is a a prime?

- **CLIQUE**
- **Instance**: A graph $G=(V, E)$ and a positive integer k.
- **Question**: Does G contain a clique of size k or more?

- **3-DIMENSIONAL MATCHING (3DM)**
- **Instance**: A set $M \subseteq W \times X \times Y$, where W, X, and Y are disjoint sets having the same number q of elements.
- **Question**: Does M contain a *matching*, that is, a subset $M' \subseteq M$ such that $|M'|=q$ and no two elements of M' agree in any coordinate?

P problems

- A problem Π belongs to the class of P if for it there is a *polynomial* algorithm.
- On the right are three known P problems.

- **REACHABILITY**
- **Instance**: A graph $G=(V, E)$ and two vertices u and v in V.
- **Question**: Does G contain a path from u to v?

- **MINIMUM TREE**
- **Instance**: A graph $G=(V, E)$, a weight function W mapping each $e \in E$ to a integer $W(e) \in Z^+$, and an integer $k \in Z^+$.
- **Question**: Does G contains a tree of size k or less?

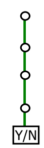
- **2-DIMENSIONAL MATCHING (2DM)**
- **Instance**: A set $M \subseteq X \times Y$, where X and Y are disjoint sets having the same number q of elements.
- **Question**: Does M contains a *matching*?

NP problems

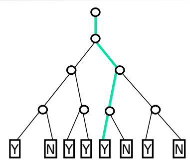
A problem Π belongs to the class of *NP* if for it there exists a *nondeterministic polynomial* algorithm

Nondetermistic algorithm:
- At some points, the algorithm faces several choices.
- The algorithm does not specify which choice the next step should take.
- A combination of the choices is called a computation (path).

Nondetermistic Alg. *A* for Π :
- For a "yes" instance, there is at least one computation leading to "yes";
- For a "no" instance, all the possible combinations lead to "no".



Deterministic Alg.
- Linear structured,
- Single outcome.



Nondeterministic Alg.
- Tree-like structure,
- Multiple outcomes.

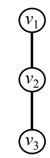
Nondetermistic polynomial Alg.
- Every computation completes within polynomial time

CLIQUE ∈ NP

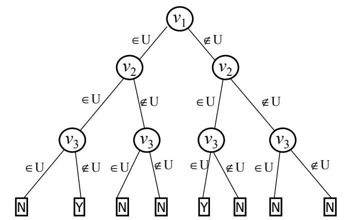
- **CLIQUE**
- **Instance**: A graph *G*=(*V*, *E*) and a positive integer *k*.
- **Question**: Does *G* contain a clique of size *k* or more?

NP Alg. for CLIQUE

U=Φ
For each v∈V do
    Let U=U+{v} or U=U
EndFor
If |U|≥k and U forms a clique
    return YES
Else
    Return NO
EndIf



Graph G



NP computation on the left graph G

SAT∈ NP

- **SATISFIABILITY (SAT)**
- **Instance**: A set *U* of Boolean variables and a collection *C* of clauses over *U* such that they form a Boolean formula Ψ in conjunction form.
- **Question**: Does a satisfying assignment for Ψ exist?

NP Alg. for CLIQUE

For each $u \in U$ do
    Let $t(u)=T$ or F
EndFor
If t satisfies Ψ
    return YES
Else
    Return NO
EndIf

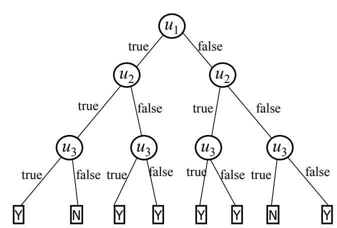
An instance

$U=\{u_1, u_2, u_3\}$, $U=\{C_1, C_2\}$ where

$C_1=\{u_1, u_2, \bar{u}_3\}$ and $C_2=\{\bar{u}_1, \bar{u}_2, u_3\}$.

That is,

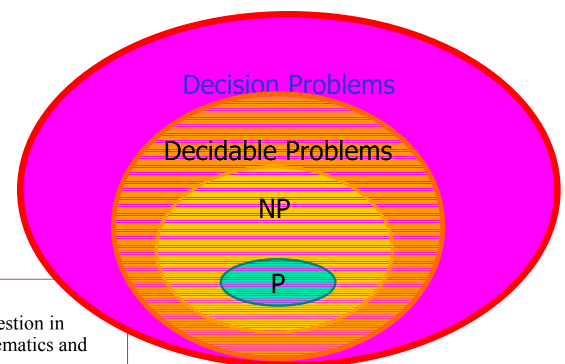
$\psi=(u_1 \cup u_2 \cup \bar{u}_3) \cap (\bar{u}_1 \cup \bar{u}_2 \cup u_3)$.



NP Alg. on the instance

P vs. NP

P⊆NP:
- All the P problems are NP problem.
- The deterministic algorithm is a special kind of the nondeterminisctic algorithm.



P=NP?
- A foremost open question in contemporary mathematics and computer science.
- Needs a major breakthrough.

Another definition for NP algorithms

An NP algorithm has two phases:

1. Guess a structure.
2. Based on the guess, compute and answer YES or NO in polynomial time of the input instance.

Notations: D_{Π}: the set of all instances for probelem Π; Y_{Π}: the set of yes-instances for Π; N_{Π}: that of no-instances. Obviously we have $D_{\Pi} = Y_{\Pi} + N_{\Pi}$.

An NP algorithm *A* solves problem Π:

- For any YES instance $I \in Y_{\Pi}$, there exists a guess, such that in the second phase the algorithm will reply with YES in polynomial time.
- For any NO instance $I \in N_{\Pi}$, every guess will lead the second phase of the algorithm *A* to reply NO in polynomial time.

In other words:

- For very $I \in D_{\Pi}$ there is a guess that makes algorithm *A* to answer YES in polynomial time if and only if $I \in Y_{\Pi}$.

NP Problems

SAT∈NP:

1. Guess an truth assignment *t*.
2. Verify whether or not *t* satisfies all the clauses (polynomial time doable).

CLIQUE∈NP:

1. Guess a subset U⊆V.
2. Verify whether |U|≥k and every pair of vertices in U is connected by an edge.

We are not sure that PRIME∈NP, but its complementary problem COMPOSITE is really in NP (COMPOSITE is to ask whether a given number k is a composite number):

1. Guess a number j that is no less than 2 and no more than k.
2. If k is dividable by j answer YES, otherwise NO.

NP-complete Problems

Intuitively, NP-complete problems are the most intractable problems in NP. A formal definition for it needs to introduce the concept of *reduction*.

Turing reduction:

Given two problems Π' and Π, if there there is an algorithm A' for Π' that uses an algorithm for Π as subroutine then we say that Π' is Turing reducible to Π.

Karp reduction:

Given two problems Π' and Π, if there there is transformation (function) $T: D_{\Pi'} \to D_{\Pi}$ such that $I \in Y_{\Pi'}$ if and only if $T(I) \in Y_{\Pi}$, then we term T a reduction (transformation) from Π' to Π.

Polynomial reduction:

A Karp reduction *T* that transforms Π' to Π is termed a polynomial reduction (transformation) if *T* can be computed in polynomial time.

We use $\Pi' \propto \Pi$ to denote Π' is polynomial reducible to Π.

NP-completeness:

$\Pi \in \mathrm{NPC} \Leftrightarrow \Pi \in \mathrm{NP}$ and $\Pi' \propto \Pi$ for all $\Pi \in \mathrm{NP}$.

How to prove NP-completeness of Π

Obviously, once an NP-complete problem has a polynomial time algorithm, all NP problems will have.

By the definition: Proving that for every $\Pi' \in \mathrm{NPC}$, there is a polynomial transformation $f_{\Pi'}$ transforming Π' to Π.

By polynomial transformation from a known NPC problem (This is correct because $\Pi_1 \propto \Pi_2$ and $\Pi_2 \propto \Pi_3 \Rightarrow \Pi_1 \propto \Pi_3$):

1. showing that $\Pi \in \mathrm{NP}$,
2. Selecting a know NP-complete problem $\Pi' \in \mathrm{NPC}$,
3. Constructing a transformation *f* from Π' to Π, and
4. Proving that *f* is a polynomial transformation:
   1) showing that *f* is computable in polynomial time, and
   2) Proving that $I \in Y_{\Pi'} \Leftrightarrow f(I) \in Y_{\Pi}$.

The first method need to do thing in a high abstract way.

The second is much easier, provided that we have known NPC problems.

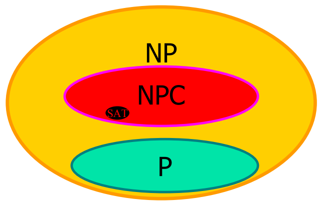
To use the second, we need a seed.

SAT, the seed for NP-completeness

Cook's theorem (1971):
SAT is NP-complete.
Proof: Omitted (S.A. Cook, Proc. 3rd Ann. ACM Symp. on Theory of Computing, pp.151-158).

Take SAT as a seed, we will breed (cultivate) following six basic NP-complete problems.

- **3-SATISFIABILITY (3SAT)**
- **Instance**: Collection $C=\{c_1, c_2, \dots, c_m\}$ of clauses on a set $U=\{u_1, u_2, \dots, u_n\}$ of Boolean variables such that $|c_i|=3$ for $1\le i\le m$.
- **Question**: Is there a truth assignment for U that satisfies all the clauses in C?



- **3-DIMENSIONAL MATCHING (3DM)**
- **Instance**: A set $M \subseteq W \times X \times Y$, where W, X, and Y are disjoint sets having the same number q of elements.
- **Question**: Does M contain a *matching*, that is, a subset $M' \subseteq M$ such that $|M'|=q$ and no two elements of M' agree in any coordinate?

Basic NPC Problems

- **VERTEX COVER (VC)**
- **Instance**: A graph $G=(V, E)$ and a positive integer k.
- **Question**: Is there a vertex cover of size k or less for G, that is, a subset $V' \subseteq V$ such that $|V'|\le k$ and for each edge $\{u,v\}\in E$, at least one of u and v belongs to V'?

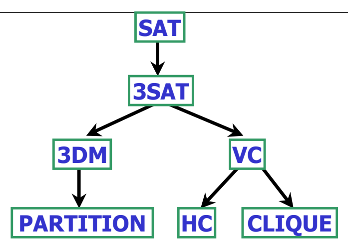
- **CLIQUE**
- **Instance**: A graph $G=(V, E)$ and a positive integer k.
- **Question**: Does G contain a clique of size k or more, that is, a subset $V' \subseteq V$ such that $|V'|\ge k$ and every two vertices in V' are joined by an edge in E?

- **HAMILTONIAN CIRCUIT (HC)**
- **Instance**: A graph $G=(V, E)$.
- **Question**: Does G contain a Hamiltonian circuit, that is, an ordering $\langle v_1, v_2, \dots, v_n\rangle$ of the vertices of G, where $n=|V|$, such that $\{v_n, v_1\}\in E$ and $\{v_i, v_{i+1}\}\in E$ for all i, $1 \le i < n$?

Basic NPC Problems

- **PARTITION**
- **Instance**: A finite set $A=\{a_1, a_2, \dots, a_n\}$ of positive integers.
- **Question**: Is there a subset $A' \subseteq A$ such that $\sum_{a\in A'} a = \sum_{a\in A-A'} a$?

On the left is the diagram of the sequence of transformations used to prove the NP-completeness of the six basic problems.



3SAT is NP-complete (I)

- 3SAT$\in$NP because we can guess a truth assignment for U and then check in polynomial time whether that truth setting satisfies all the given three-literal clauses.
- The transformation from SAT: Let $U=\{u_1, u_2, \dots, u_n\}$ and $C=\{c_1, c_2, \dots, c_m\}$ be any instance of SAT, then the C' and U' of 3SAT are

$$U' = U \cup \left[\bigcup_{1\le j\le m} U'_j\right] \text{ and } C' = \bigcup_{1\le j\le m} C'_j.$$

Thus we need only to show how C'_j and U'_j can be constructed from c_j.

Let $c_j=\{z_1, z_2, \dots, z_k\}$, where each z_i takes either the positive form of u_l or the negative form of u_l. Then the way to construct C'_j and U'_j depends on k.

Case 1. k=1: $U'_j = \{y^1_j, y^2_j\}$ and

$$C'_j = \{\{z_1, y^1_j, y^2_j\}, \{z_1, y^1_j, \bar{y}^2_j\}, \{z_1, \bar{y}^1_j, y^2_j\}, \{z_1, \bar{y}^1_j, \bar{y}^2_j\}\}$$

Case 2. k=2:

$$U'_j = \{y^1_j\} \text{ and } C'_j = \{\{z_1, z_2, y^1_j\}, \{z_1, z_2, \bar{y}^1_j\}\}$$

Case 3. k=3: $U'_j = \phi$ and $C'_j = \{c_j\}$

Case 4. k>3:

$$U'_j = \{y^i_j \mid 1 \le i \le k-3\} \text{ and } C'_j = \{\{z_1, z_2, y^1_j\}\} \cup$$
$$\{\{\bar{y}^i_j, z_{i+2}, y^{i+1}_j\} \mid 1 \le i \le k-4\} \cap \{\{\bar{y}^{k-3}_j, z_{k-1}, z_k\}\}$$

3SAT is NP-complete (II)

- To prove that the construction is indeed a transformation, we must show that the set C' of clauses is satisfiable if and only if C is.

  IF part: Suppose t: $U \rightarrow \{T, F\}$ is a truth assignment satisfying C. We extend t to U' to get t' as follows.

  Case 1 or 2: The clauses in C'_j are already satisfied by t, so we can arbitrarily extend t' to U'_j, say by setting $t'(y)=T$ for all $y \in U'_j$.

  Case 3: A trivial case.

  Case 4: Since t satisfies C_j, there must be a least number l such that $t(z_l)=T$.

l=1 or 2: Set $t'(y^i_j)=T$ for $1 \le i \le k$-3, which satisfies all the clauses of C'_j.

l=k-1 or k: Set $t'(y^i_j)=F$ for $1 \le i \le k$-3, which satisfies all the clauses of C'_j.

Otherwise, $3 \le l \le k$-2:

Set $t'(y^i_j)=T$ for $1 \le i \le l$-2, the first l-2 clauses of C'_j are satisfied.

Set $t'(y^i_j)=F$ for l-1$\le i \le k$-3, all of the l-th clause to the (k-3)-th clause of C'_j are satisfied.

The $(l-1)th$ clause $\{\bar{y}_j^{l-2}, z_l, y_j^{l-1}\}$ of C'_j has been satisfied by $t(z_l)=T$.

These choices for t' will insure that C'_j is satisfied and so C' is, by t'.

3SAT is NP-complete (III)

ONLY-IF part: Conversely, if t' is a satisfying truth assignment for , it is easy to verify that the restriction of t' to the variables in U must be a satisfying truth assignment for C.

- **The transformation can be performed in polynomial time**: To see this, it suffices to observe that the number of three-literal clauses in C' is bounded by a polynomial in mn.

The whole proof for the NP-completeness of 3SAT is finished.