Computational Complexity
计算复杂性

张方国

中山大学计算机学院

isszhfg@mail.sysu.edu.cn

Lecture 7. NP-Completeness(Cont.)

- Other NPC problems

  Set Cover is NP-complete, Clique is NP-complete,

- NP sets that are neither in P nor NP-complete
- Three relatively advanced topics in P and NP
  - Promise Problems
  - Optimal search algorithms for NP
  - The class coNP and its intersection with NP

Some natural NP-complete problems

Proposition

SAT is in P if and only if P = NP.

Proposition

If A is in NP and SAT can be reduced to A then A is NP -complete.

To prove a problem is NPC, we need to do is:

a, show that the problem is in NP,

b, reduce an NP-complete problem to it, and

c, show that the reduction is a polynomial time function.

Set Cover is NP-complete

Set cover problem: a collection of finite sets $S_1, \ldots, S_m$ and an integer K, the question (for decision) is whether or not there exist (at most) K sets that cover $\bigcup_{i=1}^{m} S_i$ (i.e., indices $i_1, \ldots, i_K$ such that $\bigcup_{j=1}^{K} S_{j_i} = \bigcup_{i=1}^{m} S_i$.

Proposition

Set Cover is NP-complete.

Proof Sketch: We sketch a reduction of SAT to Set Cover. For a CNF formula ϕ with m clauses and n variables, we consider the sets $S_{1,\mathtt{t}}, S_{1,\mathtt{f}}, .., S_{n,\mathtt{t}}, S_{n,\mathtt{f}} \subseteq \{1, ..., m\}$ such that $S_{i,\mathtt{t}}$ (resp., $S_{i,\mathtt{f}}$) is the set of the indices of the clauses (of ϕ) that are satisfied by setting the i^{th} variable to `true` (resp., `false`). That is, if the i^{th} variable appears unnegated (resp., negated) in the j^{th} clause then $j \in S_{i,\mathtt{t}}$ (resp., $j \in S_{i,\mathtt{f}}$). Note that the union of these $2n$ sets equals $\{1, ..., m\}$. Now, on input ϕ, the reduction outputs the Set Cover instance $f(\phi) \stackrel{\text{def}}{=} ((S_1, .., S_{2n}), n)$, where $S_{2i-1} = S_{i,\mathtt{t}} \cup \{m + i\}$ and $S_{2i} = S_{i,\mathtt{f}} \cup \{m + i\}$ for $i = 1, ..., n$.

Note that f is computable in polynomial-time, and that if ϕ is satisfied by $\tau_1 \cdots \tau_n$ then the collection $\{S_{2i-\tau_i} : i = 1, ..., n\}$ covers $\{1, ..., m + n\}$. Thus, $\phi \in SAT$ implies that $f(\phi)$ is a yes-instance of `Set Cover`. On the other hand, each cover of $\{m + 1, ..., m + n\} \subset \{1, ..., m + n\}$ must include either S_{2i-1} or S_{2i} for each i. Thus, a cover of $\{1, ..., m + n\}$ using n of the S_j's must contain, for every i, either S_{2i-1} or S_{2i} but not both. Setting τ_i accordingly (i.e., $\tau_i = 1$ if and only if S_{2i-1} is in the cover) implies that $\{S_{2i-\tau_i} : i = 1, ..., n\}$ covers $\{1, ..., m\}$,

Vertex Cover is NP-complete

Vertex Cover problem: which is a special case of the Set Cover problem. The instances consists of pairs (G, K) where $G = (V, E)$ is a simple graph and K is an integer, and the problem is whether or not there exists a set of (at most) K vertices that is incident to all graph edges (i.e., each edge in G has at least one endpoint in this set). Indeed, this instance of Vertex Cover can be viewed as an instance of Set Cover by considering the collection of sets $(S_v)_{v \in V}$ where S_v denotes the set of edges incident at vertex v.

Clique is NP-complete

令V'为无向图G = (V, E)顶点V的子集，当且仅当对于V'中的任意顶点u和v，(u,v)是图的一条边时，V'定义了一个完全子图。一个团就是图G的一个完全子图。一个团的大小就是该团所含顶点的数目。

以最优化形式描述的团(Clique)问题：给定一个无向图G = (V, E)，问题就是在图G中找一个团，使其所含的顶点数最多。

CLIQUE. Given a graph and an integer k, are there k vertices in the graph which are all adjacent to each other?

Proof. It is easy to verify that a graph has a clique of size k if we guess the vertices forming the clique. We merely examine the edges. This can be done in polynomial time.

We shall now reduce 3-SAT to CLIQUE. We are given a set of k clauses and must build a graph which has a clique if and only if the clauses are satisfiable. The literals from the clauses become the graph's vertices. And collections of true literals shall make up the clique in the graph we build. Then a truth assignment which makes at least one literal true per clause will force a clique of size k to appear in the graph. And, if no truth assignment satisfies all of the clauses, there will not be a clique of size k in the graph.

To do this, let every literal in every clause be a vertex of the graph we are building. We wish to be able to connect true literals, but not two from the same clause. And two which are complements cannot both be true at once. So, connect all of the literals which are not in the same clause and are not complements of each other. We are building the graph G = (V, E) where:

$$V = \{< x, i > | xisinthei - thclause\}$$

$$E = \{(< x, i >, < y, j >) | x \neq yandi \neq j\}$$

Now we shall claim that if there were k clauses and there is some truth assignment to the variables which satisfies them, then there is a clique of size k in our graph. If the clauses are satisfiable then one literal from each clause is true. That is the clique. Why? Because a collection of literals (one from each clause) which are all true cannot contain a literal and its complement. And they are all connected by edges because we connected literals not in the same clause (except for complements).

On the other hand, suppose that there is a clique of size k in the graph. These k vertices must have come from different clauses since no two literals from the same clause are connected. And, no literal and its complement are in the clique, so setting the truth assignment to make the literals in the clique true provides satisfaction.

- 一个无向图$G=(V,E)$中一个大小为k的团集是一个G中k个顶点的完全子图
- 团集问题：给定G和正整数k, G包含一个大小为k的团集吗？

团集问题CLIQUE 是NPC

证明：（1）CLIQUE$\in$NP。假设给定大小为k的集合$V'\subseteq V$，那么我们可以在多项式时间内检查V'中每一对顶点之间的边是否属于E。

（2）3-CNF-SAT$\leq_p$ CLIQUE。即设计一个算法将任何一个3-CNF的实例ϕ映射到一个图G，使得ϕ可满足当且仅当G有一个大小为k的团集。

（2）假设给定3-CNF：$\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$

每个从句 $C_r = l_1^r \vee l_2^r \vee l_3^r,\ 1 \le r \le k$

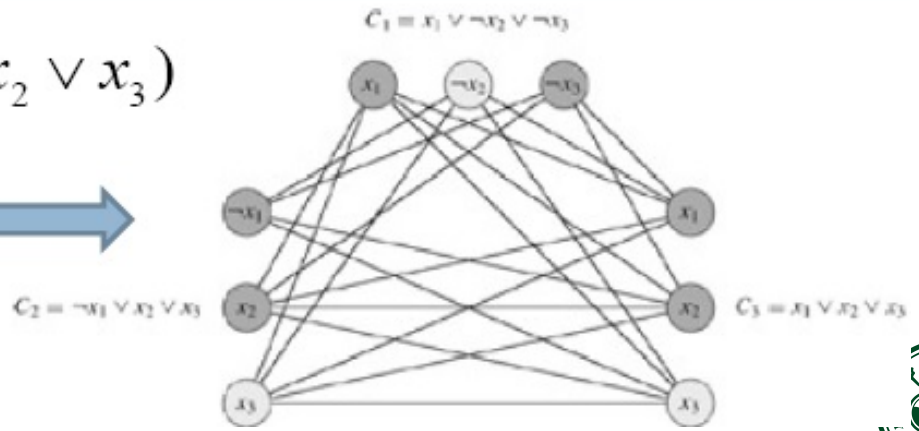
按如下方法构造图G：对 每个从句 C_r，设计3个顶点 V_1^r, V_2^r, V_3^r.

两个顶点若满足下列两条性质，则以边相连 $(V_i^r, V_j^s) \in E$

1. $r \neq s$
2. l_i^r and l_j^s are consistent, i.e. l_i^r is not the negation of l_j^s

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

构造图G的时间只需要多项式时间



- ϕ 可满足当且仅当G有一个大小为k的团集

证明：$\Rightarrow$假设 ϕ 有一组变量赋值使得输出为1，那么每个从句中 C_r 至少有一个文字 l_i^r 被赋为1。那么将其对应的G中的顶点 v_i^r 选中，从而得到一个大小为k的顶点集合V′。下面证明V′是一个团集，即任意两个顶点都有边相连。V′中任意两个顶点 v_i^r, v_j^s，$r \neq s$，并且其相应的文字 l_i^r, l_j^s 都赋值为1，所以不会是非的关系。所以，边 $(v_i^r, v_j^s) \in E$。

$\Leftarrow$ 如果G有一个大小为k的团集V′。由于G中没有连接同一个从句中所对应的变量，所以V′中正好包含每一个从句中所对应的一个变量，记这个变量对应的文字为 l_i^r，并且赋值为1。由于G中边不会连接一个文字和它的非，所以不用担心这样赋值会自相矛盾。因此 ϕ 中每一个从句都可以满足，从而 ϕ 可以满足。（对于 ϕ 中不对应团集V′中顶点的文字，可任意赋值）

Other NP-complete Problems

Graph 3-Colorability is NP-complete.

reduce 3-SAT to 3-Colorability.

A systems of linear equations over $GF(2)$ and an integer k, and the problem is to determine whether there exists an assignment that satisfies at least k equations.

A system of cubic equations over $GF(2)$, the problem is to determine whether there exists an assignment that satisfies all the equations.

by a reduction from 3SAT that maps the clause $x \vee \neg y \vee z$ to the equation $(1-x) \cdot y \cdot (1-z) = 0$.

A system of quadratic equations over $GF(2)$ and the problem is to determine whether there exists an assignment that satisfies all the equations.

0-1 Integer Programming (0-1 INT). Given a matrix A and a vector b, is there a vector x with values from $\{0, 1\}$ such that $Ax \geq b$?

If we did not require the vector x to have integer values, then this is the linear programming problem and is solvable in polynomial time. This one is more difficult.

Theorem. 0-1 INT is NP-complete.

Here are some of the more commonly known problems that are NP-complete when expressed as decision problems. This list is in no way comprehensive (there are more than 3000 known NP-complete problems). Most of the problems in this list are taken from Garey and Johnson's seminal book Computers and Intractability: A Guide to the Theory of NP-Completeness, and are here presented in the same order and organization.

http: //en.wikipedia.org / wiki / List of NP-complete problems

NP sets that are neither in P nor NP-complete

Any NP-set to be either NP-complete or in P? NO!

Theorem

Assuming $\mathcal{NP} \neq \mathcal{P}$ there exist a set T in $\mathcal{NP} \setminus \mathcal{P}$ such that some sets in $\mathcal{NP}$ are not Cook-reducible to T(such class denote $\mathcal{NPI}$).

Ladner's Theorem: Existence of NP-Intermediate problems.

$$SAT_H = \{\psi 01^{n^{H(n)}} : \psi \in SAT \ and \ n = |\psi|\}.$$

$H : N \to N$定义如下:

$H(n)$ is the smallest number $i < \log\log n$ such that for every $x \in \{0,1\}^*$ with $|x| \leq \log n$.

Three relatively advanced topics in P and NP

- Promise Problems
- Optimal search algorithms for NP
- The class coNP and its intersection with NP

Promise Problems

Promise problems are a natural generalization of search and decision problems, where one explicitly considers a set of legitimate instances(rather than considering any string as a legitimate instance).

Even, Selman and Yacobi introduced promise problems(S. Even, A.L. Selman and Y. Yacobi, The Complexity of Promise Problems with Applications to Public Key Cryptography. Information and Control, Vol.61, pages159-173, 1984).

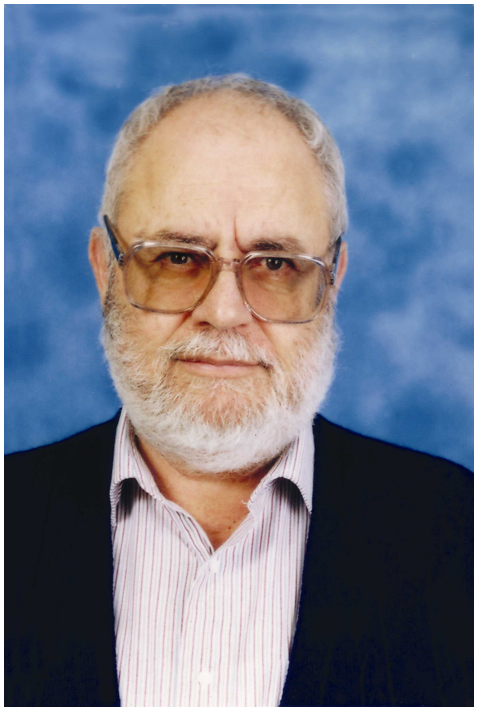


Figure: Shimon Even [1935-2004]

In the context of search problems, a promise problem is a relaxation in which one is only required to find solutions to instances in a predetermined set, called the promise

search problems with a promise: A search problem with a promise consists of a binary relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ and a promise set P. Such a problem is also referred to as the search problem R with promise P.

- The search problem R with promise P is solved by algorithm A if for every $x \in P$ it holds that $(x, A(x)) \in R$ if $x \in S_R = \{x : R(x) \neq \emptyset\}$ and $A(x) = \perp$ otherwise, where $R(x) = \{y : (x,y) \in R\}$.

  The time complexity of A on inputs in P is defined as $T_{A|P}(n) = max_{x \in P \cap \{0,1\}^n}\{t_A(x)\}$, where $t_A(x)$ is the running time of $A(x)$ and $T_{A|P}(n) = 0$ if $P \cap \{0,1\}^n = \emptyset$.

- The search problem R with promise P is in the promise problem extension of $\mathcal{PF}$ if there exists a polynomial-time algorithm that solves this problem.
- The search problem R with promise P is in the promise problem extension of $\mathcal{PC}$ if there exists a polynomial T and an algorithm A such that, for every $x \in P$ and $y \in \{0,1\}^*$, algorithm A makes at most $T(|x|)$ steps and it holds that $A(x,y) = 1$ if and only if $(x,y) \in R$.

These are search problem in which the promise is that the instance has a solution (i.e., in terms of the foregoing notation $P = S_R$). We refer to such search problems by the name candid search problems.

Definition

(candid search problems:) An algorithm A solves the candid search problem of the binary relation R if for every $x \in S_R = \{x : \exists y \ s.t. (x, y) \in R\}$ it holds that $(x, A(x)) \in R$. The time complexity of such an algorithm is defined as $T_{A|S_R}(n) = max_{x \in S_R \cap \{0,1\}^n}\{t_A(x)\}$, where $t_A(x)$ is the running time of $A(x)$ and $T_{A|S_R}(n) = 0$ if $S_R \cap \{0,1\}^n = \emptyset$.

Decision problems with a promise: In the context of decision problems, a promise problem is a relaxation in which one is only required to determine the status of instances that belong to a predetermined set, called the promise.

Definition

A promise problem consists of a pair of non-intersecting sets of strings, denoted (S_{yes}, S_{no}) and $S_{yes} \cup S_{no}$ is called the promise.

- The promise problem (S_{yes}, S_{no}) is solved by algorithm A if for every $x \in S_{yes}$ it holds that $A(x) = 1$, and for every $x \in S_{no}$ it holds that $A(x) = 0$. The promise problem is in the promise problem extension of $\mathcal{P}$ if there exists a polynomial-time algorithm that solves it.

- The promise problem (S_{yes}, S_{no}) is in the promise problem extension of $\mathcal{NP}$ if there exists a polynomial p and a polynomial-time algorithm V such that the following two conditions hold:
  - Completeness: For every $x \in S_{yes}$ there exists y of length at most $p(|x|)$, such that $V(x, y) = 1$.
  - Soundness: For every $x \in S_{no}$ and every y, it holds that $V(x, y) = 0$.

If the promise is efficiently recognizable, then we may avoid mentioning the promise by using one of the following two conventions:

- Extending the set of instances to the set of all possible strings
- Considering every string as a valid encoding of an object that satisfies the promise

Optimal search algorithms for NP

Candid search problem of any relation in $\mathcal{PC}$: $\mathcal{PC}$, candid search problem(which the solver is promised the given instance has a solution).

Theorem

For every binary relation $R \in \mathcal{PC}$ there exists an algorithm A that satisfies the following:

- *A solves the candid search problem of R.*
- *There exists a polynomial p such that for every algorithm A' that solves the candid search problem of R and for every $x \in S_R = \{x : R(x) \neq \emptyset\}$ it holds that $t_A(x) = O(t_{A'}(x) + p(|x|))$, where t_A (resp., $t_{A'}$) denotes the number of steps taken by A (resp., A') on input x*

The class coNP and its intersection with NP

$$co\mathcal{C} \stackrel{\text{def}}{=} \{\{0,1\}^* \setminus S : S \in \mathcal{C}\}$$

$$co\mathcal{NP} \stackrel{\text{def}}{=} \{\{0,1\}^* \setminus S : S \in \mathcal{NP}\}$$

If $R \in \mathcal{PC}$, then $S_R = \{x : \exists y \ s.t. (x,y) \in R\}$ is in $\mathcal{NP}$. It follows that the set of instances having no solution (i.e., $\{0,1\}^* \setminus S_R = \{x : \forall y \ (x,y) \notin R\}$) is in $co\mathcal{NP}$.

It is widely believed that $\mathcal{NP} \neq co\mathcal{NP}$, this conjecture implies $\mathcal{P} \neq \mathcal{NP}$.

The conjecture $\mathcal{NP} \neq co\mathcal{NP}$ means that some sets in $co\mathcal{NP}$ do not have NP-proof systems (because $\mathcal{NP}$ is the class of sets having NP-proof systems).

Under this conjecture, the complements of NP-complete sets do not have NP-proof systems, i.e.,

Theorem

Suppose that $\mathcal{NP} \neq co\mathcal{NP}$ and let $S \in \mathcal{NP}$ such that every set in $\mathcal{NP}$ is Karp-reducible to S. Then $\overline{S} \stackrel{def}{=} \{0,1\}^* \setminus S$ is not in $\mathcal{NP}$.

Proof Sketch: We first observe that the fact that every set in $\mathcal{NP}$ is Karp-reducible to S implies that every set in co$\mathcal{NP}$ is Karp-reducible to $\overline{S}$. We next claim that *if S' is in $\mathcal{NP}$ then every set that is Karp-reducible to S' is also in $\mathcal{NP}$*. Applying the claim to $S' = \overline{S}$, we conclude that $\overline{S} \in \mathcal{NP}$ implies co$\mathcal{NP} \subseteq \mathcal{NP}$, which in turn implies $\mathcal{NP} =$ co$\mathcal{NP}$ in contradiction to the main hypothesis.

We now turn to prove the foregoing claim; that is, we prove that if S' has an NP-proof system and S'' is Karp-reducible to S' then S'' has an NP-proof system. Let V' be the verification procedure associated with S', and let f be a Karp-reduction of S'' to S'. Then, we define the verification procedure V'' (for membership in S'') by $V''(x, y) = V'(f(x), y)$. That is, any NP-witness that $f(x) \in S'$ serves as an NP-witness for $x \in S''$ (and these are the only NP-witnesses for $x \in S''$). This may not be a "natural" proof system (for S''), but it is definitely an NP-proof system for S''. □

Theorem

If every set in $\mathcal{NP}$ can be Cook-reduced to some set in $\mathcal{NP} \cap co\mathcal{NP}$ then $\mathcal{NP} = co\mathcal{NP}$.

In particular, assuming $\mathcal{NP} \neq co\mathcal{NP}$, no set in $\mathcal{NP} \cap co\mathcal{NP}$ can be NP-complete, even when NP-completeness is defined with respect to Cook-reductions. Since $\mathcal{NP} \cap co\mathcal{NP}$ is conjectured to be a proper superset of $\mathcal{P}$, it follows (assuming $\mathcal{NP} \neq co\mathcal{NP}$) that there are decision problems in $\mathcal{NP}$ that are neither in $\mathcal{P}$ nor NP-hard (i.e., specifically, the decision problems in $\mathcal{NP} \cap co\mathcal{NP} \setminus \mathcal{P}$)

Recall that on top of the $\mathcal{P} \neq \mathcal{NP}$ conjecture, we mentioned two other conjectures(which clearly imply $\mathcal{P} \neq \mathcal{NP}$)

- The conjecture that$\mathcal{NP} \neq co\mathcal{NP}$(equivalently, $\mathcal{NP} \cap co\mathcal{NP} \neq \mathcal{NP}$).

  This conjecture is equivalent to the conjecture that CNF formulae have no short proofs of unsatisfiability.
- The conjecture that $\mathcal{NP} \cap co\mathcal{NP} \neq \mathcal{P}$

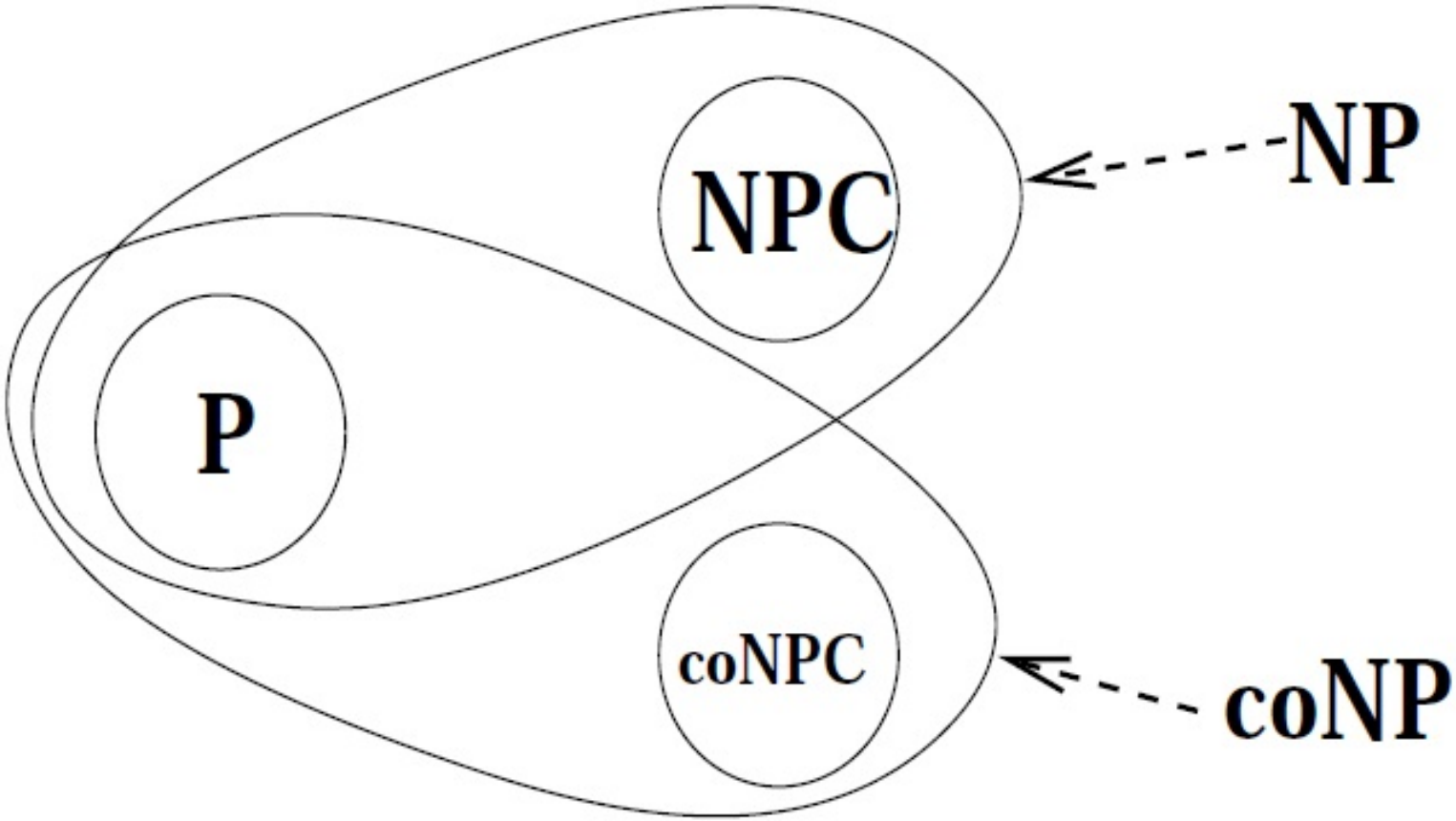


Figure 2.5: The world view under $\mathcal{P} \neq co\mathcal{NP} \cap \mathcal{NP} \neq \mathcal{NP}$.

Definition

A set S is called $co\mathcal{NP}-$hard if every set in $co\mathcal{NP}$ is Karp-reducible to S. A set is called $co\mathcal{NP}-$complete if it is both in $co\mathcal{NP}$ and $co\mathcal{NP}-$hard.

Thank You Very Much!