NP != P

By Liu Ran

Table of Contents

1. Introduce

The P versus NP problem is a major unsolved problem in computer science. Informally, it asks whether a computer can also quickly solve every problem whose solution can be quickly verified by a computer. It was introduced in 1971 by Stephen Cook in his seminal paper "The complexity of theorem proving procedures" and is considered by many to be the most important open problem in the field.

The informal term quickly used above means the existence of an algorithm for the task that runs in polynomial time. The general class of questions for which some algorithm can provide an answer in polynomial time is called "class P" or just "P". For some questions, there is no known way to find an answer quickly, but if one is provided with information showing what the answer is, it may be possible to verify the answer quickly. The class of questions for which an answer can be verified in polynomial time is called NP.

NP-complete problems are a set of problems to each of which any other NP-problem can be reduced in polynomial time, and whose solution may still be verified in polynomial time. That is, any NP problem can be transformed into any of the NP-complete problems. Informally, an NP-complete problem is at least as "tough" as any other problem in NP.

NP-hard problems are those at least as hard as NP-complete problems, i.e., all NP-problems can be reduced (in polynomial time) to them. NP-hard problems need not be in NP, i.e., they need not have solutions verifiable in polynomial time.

In information theory, entropy is a measure of the uncertainty in a random variable. In this context, the term usually refers to the Shannon entropy, which quantifies the expected value of the information contained in a message. Entropy is typically measured in bits, nats, or bans. Shannon entropy is the average unpredictability in a random variable, which is equivalent to its information content.

Claude Elwood Shannon (April 30, 1916 – February 24, 2001) was an American mathematician, electronic engineer, and cryptographer known as "The father of information theory".

Shannon is famous for having founded information theory with a landmark paper that he published in 1948.

The Shannon entropy, a measure of uncertainty (see further below) and denoted by *H(x)*, is defined by Shannon as

$$H(x)=E[I(x_i)]=E[\,\log(2,1/p(x_i))\,]=-\sum_{i=1}^{n} p(x_i)\log(2,p(x_i)) \quad (i=1,2,..n)$$

Where $p(x_i)$ is the probability mass function of outcome x_i.

Specially, if there are N outcomes, and $p(x_1)=p(x_2)=\ldots=p(x_n)=1/N$.

$$H(x) = -\sum_{i=1}^{n} p(x_i)\log(2,p(x_i)) = -\sum_{i=1}^{n}(1/N)\log(2,(1/N)) = \sum_{i=1}^{n}(1/N)\log(2,N)$$

$$= \log(2,N).$$

2. Preliminary theorem

(2.1) Polynomial identical theorem:

f(x) = $a_k x^k + a_{k-1}x^{k-1} + ... + a_1 x + a_0, a_k \neq 0$;

g(x) = $b_k x^k + b_{k-1}x^{k-1} + ... + b_1 x + b_0, b_k \neq 0$;

f(x) = g(x) $\Leftrightarrow a_k = b_k, a_{k-1} = b_{k-1}, ..., a_1 = b_1, a_0 = b_0, a_k \neq 0$

(2.2) Binomial theorem:

Binomial theorem calls also Newton binomial theorem. Newton brought forth it in 1664-1665 year.

$$(a+b)^n = C_n^0 a^n + C_n^1 a^{n-1} b + \ldots + C_n^i a^{n-i} b^i + \ldots + C_n^n b^n, i>0, i<n$$

C_n^i expresses combinational number of taking freely i elements from n elements, $C_n^i = n!/((n-i)!i!)$

(2.3) The entropy of P problem is zero, i.e. H(P) = 0.

Because every step of P class problem is deterministic, its happen probability is always 1. Base on the definition of P problem, P problem can via polynomial steps to get 1 deterministic outcome. Define *T(n)* = *O*(n^k), we can express the happen probability as p = $\prod_{i=1}^{T(n)} p_i$, p_i=1 $\Rightarrow$ p = 1 $\Rightarrow$ there is only 1 outcome for P problem $\Rightarrow$ entropy of P problem *H(P)* = $-1 \times \log(2,1) = 0$.

(2.4) The entropy of NP problem is above zero, i.e. H(NP) > 0.

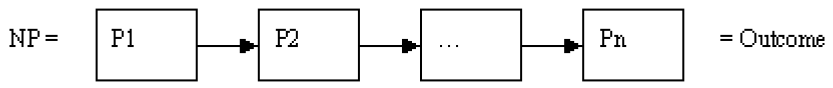
Because every step of NP class problem is non-deterministic, its happen probability is always < 1. Base on the definition of NP problem, NP problem can only via polynomial steps to verify 1 outcome is answer or not. Every step has more than 1 choice to calculate. Define *T'(n)* = *O(k^n)*,we can express the happen probability of one outcome as p(x_i) = $\prod_{i=1}^{T'(n)} p_i$, $p_i < 1 \Rightarrow$ p(x_i) < 1. Because *H(NP)* = $-\sum_{i=1}^{n} p(x_i)\log(2,p(x_i))$ = $\sum_{i=1}^{n} p(x_i)\log(2,1/p(x_i)) > \sum_{i=1}^{n} p(x_i)\log(2,1) = 0$.

Base on the definition of NP problem, NP problem is easy to verify in polynomial time. Every outcome provided is deterministic, but why H(NP) > 0? Because every outcome provided is deterministic. Every step is deterministic. But the final outcome is non-deterministic to be the correct answer. It's only one of many outcomes. Denote the happen probability of one outcome as p(x_i), the we can express the happen probability as p = p(x_i)$\times \prod_{i=1}^{T(n)} p_i$ =p(x_i)$\times$1= p(x_i)<1.

Specially, when the happen probability of every outcome is identical. If there are N outcomes, p(x_i) = 1/N. It's is deterministic to verify one outcome via polynomial steps, but happen probability of the outcome being correct answer is 1/N.

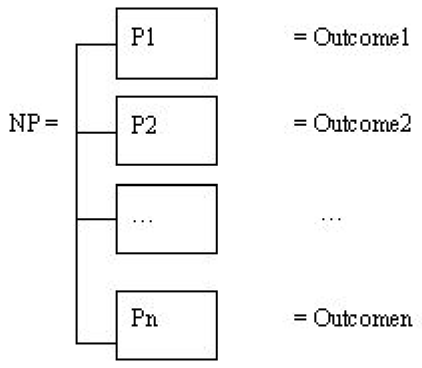
(2.5) If a NP problem can reduce to P problems, every P problem must be one of many outcomes.

(2.5.1) Suppose a NP problem can reduce to a serial of P problems.



Because every step of P problem is deterministic and the happen probability of the outcome is 1. ⇒ H(NP) = 0, which is contradictory with (2.4) H(NP) > 0, so Suppose is false. ⇒ A NP problem can't reduce to a serial of P problems.

(2.5.2) A NP problem can reduce to many parallel P problems.



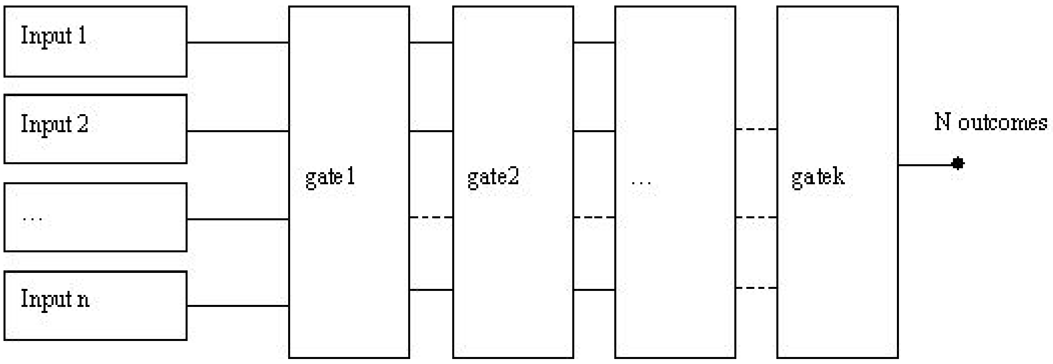
Because every step of P problem is deterministic and (2.4) H(NP) > 0 ⇒the only non-deterministic step is the outcome, ⇒ NP problem must have many outcomes and every P problem's outcome is only one of all outcomes. ⇒ A NP problem can reduce to many parallel P problems.

From (2.5.1) and (2.5.2) ⇒ A NP problem can reduce to many parallel P problems, every P problem must be one of many outcomes.

3. Proof

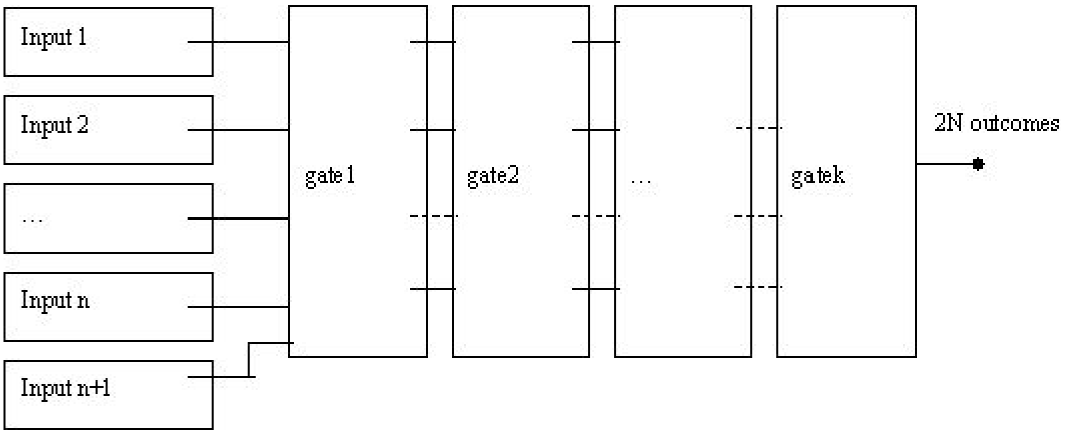
Any NP problem can be transformed into any of the NP-complete problems. The first NP-complete problem is the logic circuit. That is, if the logic circuit equal to P problem, NP = P is proven; if not equal to, NP $\neq$ P is proven.

Let's considerate a logic circuit like below.



n inputs via k gates, then output.

Every input can be value 0 or 1, suppose n inputs can generate N outcomes. Express as G(n) = N.



Input increase to n+1, because input is value 0 or 1, new out can be express as G(n+1) = ($N|_{n+1=0}$) + ($N|_{n+1=1}$) = *2N*. It means that inputs from 1 to n generate N outcomes, and input (n+1) is value 0; inputs from 1 to n generate N outcomes, and input n+1 is value 1. The total outcomes are 2N.

The logic circuit's entropy is *log(2,N)* when n inputs; the logic circuit's entropy is *log(2,2N)* when n+1 inputs. The delta entropy $\vartriangle H$ = *H(n+1) – H(N) = log(2,2N) – log(2,N*)= 1 (3.1).

(3.2) Suppose NP = P, it means that

(3.2.1) Any NP problem can reduce to one P problem in polynomial time. I.e. *NP* = P_1;

(3.2.2) Any NP problem can reduce to polynomial parallel P problem in polynomial time. I.e. $NP(n) = \sum_{i=1}^{T(n)} P_i(n)$, $T(n) = O(n^k)$;

(3.2.3) Any NP problem can reduce to exponential parallel P problem in polynomial time. I.e. $NP(n) = \sum_{i=1}^{T(n)} P_i(n)$, $T(n) = O(k^{p(n)})$;

(3.2.4) Any NP problem can reduce to more than exponential parallel P problem in polynomial time. I.e. $NP(n) = \sum_{i=1}^{T(n)} P_i(n)$, $T(n) > O(k^{p(n)})$.

For (3.2.1), if *NP* = P_1, from preliminary theorem (2.3) H(P) = 0 $\Rightarrow \vartriangle H = H(P_1(n+1)) - H(P_1(n))$= 0 – 0 = 0. It's contradictory with (3.1);

For (3.2.2), $NP(n)=\sum_{i=1}^{T(n)} P_i(n)$, *T(n) = O(*$n^k$*)*, because preliminary theorem (2.3) H(P) = 0 and (2.4) H(NP) > 0 $\Rightarrow$every P_i*(n)* is one of many outcomes, which include information quantity and reduce indeterminacy. And because (3.1)$\Rightarrow$ $\vartriangle H$ *=H(*P_1*(n+1)) – H(*P_1*(n)) = log(2, T(n+1) - log(2, T(n)=1* $\Rightarrow$ *log(2, T(n+1) - log(2, T(n) = log(2, T(n+1)/T(n)) = 1* $\Rightarrow$ *T(n+1)/T(n) = 2,* denote *T(n)* = $a_{k'}x^{k'}+a_{k'-1}x^{k'-1}+...+a_1x+a_0, a_{k'}\neq 0$ $\Rightarrow$ $T(n+1)=a_{k'}(n+1)^{k'}+a_{k'-1}(n+1)^{(k'-1)}+...+a_1(n+1)+a_0$ = *2T(n)* = *2(*$a_{k'}n^{k'}+a_{k'-1}n^{(k'-1)}+...+a_1n+a_0$*),* because of (2.1) Polynomial identical theorem and (2.2) Binomial theorem $\Rightarrow$ $2a_{k'}n^{k'}=a_{k'}n^{k'}\Rightarrow a_{k'}=0$, it's contradictory with $a_{k'}\neq 0$, *T(n) =*$a_{k'}n^{k'}+a_{k'-1}n^{(k'-1)}+...+a_1n+a_0, a_{k'}\neq 0$.

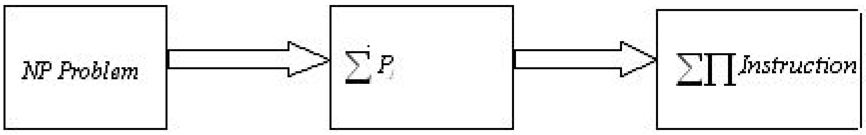
For (3.2.3), $NP(n)=\sum_{i=1}^{T(n)} P_i(n)$, *T(n) = O(* $k^{p(n)}$ *)*, because preliminary theorem (2.3) H(P) = 0 and (2.4) H(NP) > 0 $\Rightarrow$every P_i*(n)* is one of many outcomes, which include information quantity and reduce indeterminacy $\Rightarrow$ NP(n)'s complexity >= *T(n).n = O(* $k^{p(n)}$ *)).n >= O(*$k^{p(n)}$*)* $\Rightarrow$ NP(n)'s complexity is exponential, it's contradictory with NP = P.

For (3.2.4), $NP(n)=\sum_{i=1}^{T(n)} P_i(n)$, *T(n) > O(* $k^{p(n)}$ *)*, because preliminary theorem (2.3) H(P) = 0 and (2.4) H(NP) > 0 $\Rightarrow$every P_i*(n)* is one of many outcomes, which include information quantity and reduce indeterminacy $\Rightarrow$ NP(n)'s complexity >= *T(n).n > O(*$k^{p(n)}$*).n >= O(*$k^{p(n)}$*)*

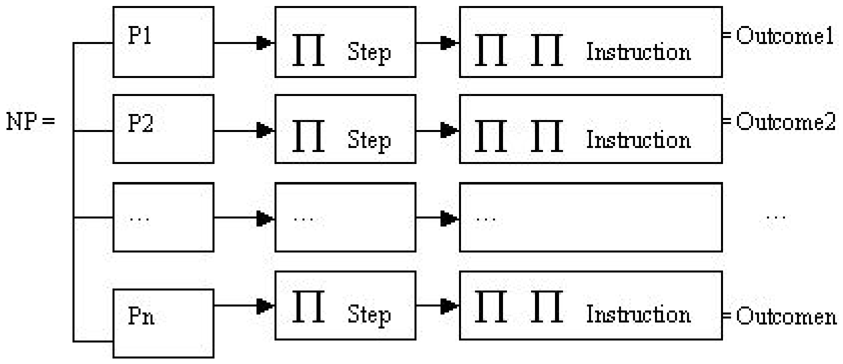
$\Rightarrow$ NP(n)'s complexity is more than exponential, it's contradictory with NP = P.

4. Explain

To explain my proof clearly, I draw a flow chart to denote that computer handles NP problem process. $\sum$ denotes parallel relationship and $\prod$ denotes serial relationship in below figures.



More detailed flow chart is below.



Any NP problem must reduce to P problem and every P problem is one of many outcomes. Any P problem must reduce to basic instruction.

But if NP=P, it violates entropy theorem. Any NP problem can't reduce to polynomial P problem.

When NP reduces to 1 P problem, the delta entropy is always zero.

When NP reduces to polynomial P problem, the delta entropy does not equal to 1.

When NP reduces to exponential P problem or more complex, the complexity has become contradictory with definition of P problem.

All scenarios are contradictory, so NP = P is wrong,

5. Conclusion

In essence, P problem is a deterministic problem, which can reduce to basic instructions in polynomial time.

NP problem is a non-deterministic problem, which can’t reduce to P problem in polynomial time.

If NP = P, it means that deterministic problem equals to non-deterministic problem, which violates information entropy principle.

So any non-deterministic problem is not easy to calculate.

References

[1] Ihara, Shunsuke (1993). Information theory for continuous systems. World Scientific. p. 2. ISBN 978-981-02-0985-8.

[2] In this context, a 'message' means a specific realization of the random variable.

[3] Brillouin, Léon (2004). Science & Information Theory. Dover Publications. p. 293. ISBN 978-0-486-43918-1.

[4] Shannon, Claude E. (July/October 1948). "A Mathematical Theory of Communication". Bell System Technical Journal27 (3): 379–423.

[5] Goise, Francois & Olla, Stefano (2008). Entropy methods for the Boltzmann equation: lectures from a special semester at the Centre Émile Borel, Institut H. Poincaré, Paris, 2001. Springer. p. 14. ISBN 978-3-540-73704-9.

[6] A b Schneier, B: Applied Cryptography, Second edition, page 234. John Wiley and Sons.

[7] A b Shannon, Claude E.: Prediction and entropy of printed English, The Bell System Technical Journal, 30:50–64, January 1951.