About the impossibility to prove P=NP or P!=NP and the pseudo-randomness in NP

Rémon, M.

Publication date:
2009

About the impossibility to prove $\boldsymbol{P \neq NP}$ or $\boldsymbol{P = NP}$ and the pseudo-randomness in $\boldsymbol{NP}$

Prof. Marcel Rémon

Abstract

The relationship between the complexity classes $\boldsymbol{P}$ and $\boldsymbol{NP}$ is an unsolved question in the field of theoretical computer science. In this paper, we look at the link between the $\boldsymbol{P}$ - $\boldsymbol{NP}$ question and the *"Deterministic"* versus *"Non Deterministic"* nature of a problem, and more specifically at the temporal nature of the complexity within the $\boldsymbol{NP}$ class of problems. Let us remind that the $\boldsymbol{NP}$ class is called the class of *"Non Deterministic Polynomial"* languages. Using the meta argument that results in Mathematics should be *"time independent"* as they are reproducible, the paper shows that the $\boldsymbol{P \neq NP}$ assertion is impossible to prove in the *a-temporal* framework of Mathematics. A similar argument based on randomness shows that the $\boldsymbol{P = NP}$ assertion is also impossible to prove, so that the $\boldsymbol{P}$ - $\boldsymbol{NP}$ problem turns out to be *"unprovable"* in Mathematics. This is not an *undecidability* theorem, as undecidability points to the paradoxical nature of a proposition. In fact, this paper highlights the time dependence of the complexity for any $\boldsymbol{NP}$ problem, linked to some pseudo-randomness in its heart.

Index Terms

Algorithm Complexity, Non Deterministic Languages, $\boldsymbol{P - NP}$ problem, 3-CNF-SAT problem

I. Introduction

A. The class $\boldsymbol{P}$ of languages

A decision problem is a problem that takes as input some string, and outputs "yes" or "no". If there is an algorithm (say a Turing machine, or a computer program with unbounded memory) which is able to produce the correct answer for any input string of length n in at

M.Rémon, Department of Mathematics, Namur University, Belgium; marcel.remon@fundp.ac.be

most $c\,n^k$ steps, where k and c are constants independent of the input string, then we say that the problem can be solved in polynomial time and we place it in the class $\boldsymbol{P}$.

More formally, $\boldsymbol{P}$ is defined as the set of all languages which can be decided by a deterministic polynomial-time Turing machine. Here we follow the framework proposed by Stephen [1]. Let Σ be a finite alphabet with at least two elements, and let Σ^* be the set of finite strings over Σ. Then a language over Σ is a subset L of Σ^*. Each Turing Machine M has an associated input alphabet Σ. For each string w in Σ^*, there is a computation associated with M, with input w. We say that M *accepts* w if this computation terminates in the accepting state "*Yes*". Note that M fails to accept w either if this computation ends in the rejecting state "*No*", or if the computation fails to terminate.

The *language accepted by* M, denoted $L(M)$, has associated alphabet Σ and is defined by

$$L(M) = \{w \in \Sigma^* | M \text{ accepts } w\}$$

We denote by $t_M(w)$ the number of steps in the computation of M on input w. If this computation never halts, then $t_M(w) = \infty$. For $n \in I\!N$, we denote by $T_M(n)$ the worst case run time of M; that is

$$T_M(n) = \max\{t_M(w) | w \in \Sigma^n\}$$

where Σ^n is the set of all strings over Σ of length n. We say that M *runs in polynomial time* if :

$$\exists k \ \in I\!N \text{ such that } \quad \{\forall n : T_M(n) \leq n^k + k \ \}$$

Definition I.1: We define the class $\boldsymbol{P}$ of languages by

$$\boldsymbol{P} = \{L | L = L(M) \text{ for a machine } M \text{ which runs in polynomial time}\}$$

B. The class $\boldsymbol{NP}$ *of languages*

The notation $\boldsymbol{NP}$ stands for *non deterministic polynomial time*, since originally $\boldsymbol{NP}$ was defined in terms of non deterministic machines. However, it is customary to give an equivalent definition using the notion of a *checking relation*, which is simply a binary relation $R \subseteq \Sigma^* \times \Sigma_1^*$ for some finite alphabets Σ and Σ_1. We associate with each such relation R a

language L_R over $\Sigma \cup \Sigma_1 \cup \{\#\}$ defined by

$$L_R = \{w\#y | R(w,y)\}$$

where the symbol $\#$ is not in Σ. We say that R is *polynomial-time* iff $L_R \in \boldsymbol{P}$.

Definition I.2: We define the class $\boldsymbol{NP}$ of languages by the condition that a language L over Σ is in $\boldsymbol{NP}$ iff there is $k \in \mathbb{N}$ and a polynomial-time checking relation R such that for all $w \in \Sigma^*$,

$$w \in L \Leftrightarrow \exists y(|y| \leq |w|^k \text{ and } R(w,y))$$

where $|w|$ and $|y|$ denote the lengths of w and y, respectively. We say that y is *a certificate* associated to w.

C. The $\boldsymbol{P}$ - $\boldsymbol{NP}$ question

The *"$\boldsymbol{P}$ versus $\boldsymbol{NP}$ problem"*, i.e. the question whether $\boldsymbol{P} = \boldsymbol{NP}$ or $\boldsymbol{P} \neq \boldsymbol{NP}$, is an open question and is the core of this paper. See [4] for the history of the question. Here, we show that neither $\boldsymbol{P} = \boldsymbol{NP}$ nor $\boldsymbol{P} \neq \boldsymbol{NP}$ can be proved in the *"a-temporal"* framework of Mathematics where results should always be reproducible. We link this assertion to the existence of some pseudo-random part in the heart of any $\boldsymbol{NP}$ problem.

D. An example of $\boldsymbol{NP}$ problem : the 3-CNF-satisfiability problem

Boolean formulae are built in the usual way from propositional variables x_i and the logical connectives $\wedge$, $\vee$ and $\neg$, which are interpreted as conjunction, disjunction, and negation, respectively. A *literal* is a propositional variable or the negation of a propositional variable, and a *clause* is a disjunction of literals. A Boolean formula is *in conjunctive normal form* iff it is a conjunction of clauses.

A *3-CNF formula* φ is a Boolean formula in conjunctive normal form with exactly three literals per clause, like $\varphi := (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) := \psi_1 \wedge \psi_2$. The *3-CNF-satisfiability or 3-CNF-SAT problem* is to decide whether there exists or not logical values for the literals so that φ can be true (on the previous example, $\varphi = 1$(True) if $x_1 = \neg x_2 = 1$).

Until now, nobody knows whether or not it is possible to check the satisfiability of any given

3-CNF formula φ in a polynomial time, as the *3-CNF-SAT* problem is known to belong to the class ***NP*** of problems. See [2] for details.

Let us give some general properties of the *3-CNF* formulae.

The *size* s of a 3-CNF formula φ is defined as the size of the corresponding *Boolean circuit*, i.e. the number of logical connectives in φ. Let us note the following property of the size s :

$$s = \mathcal{O}(m) = \mathcal{O}(n^3) \tag{1}$$

where n is the number of propositional variables x_i and m the number of clauses in φ. Indeed,

$$\frac{n}{3} \leq m \leq 2^3 \frac{n(n-1)(n-2)}{3 \times 2} \qquad \text{and} \qquad (3m-1) \leq s \leq (6m-1)$$

as there is a maximum of $2^3 \times C_3^n$ possible clauses which corresponds to the choice of 3 different variables among n, each of them being in an affirmative or negative state. Note that $s = 3m-1$ when there is no "$\neg$" in φ [$m \times 2$ logical connectives "$\vee$" for the ψ_i and $m-1$ "$\wedge$" as conjonctions] and $s = 6m-1$ when all the litterals in φ are in a negative form.

In this paper, we define the *dimension* d of a 3-CNF formula as (n, m). And we represent any 3-CNF formula by a matrix $\mathcal{A}$ of size $2n \times m$. The *signature* u_i *of a clause* ψ_i is defined as the value of the binary number corresponding to the row in the matrix. The *signature of a formula* is the ordered vector of these clause's signatures : $\varphi_{n,m} \approx (u_1, u_2, \cdots, u_m)$ with $21 \leq u_i \leq 21 \cdot 2^{2n-5}$ and $u_i > u_j$ for $i < j$. See `Table I.`

			3-CNF formula φ (*dimension* $d = (4,3)$)								
			x_1	$\neg x_1$	x_2	$\neg x_2$	x_3	$\neg x_3$	x_4	$\neg x_4$	u_i
ψ_1 :	$(x_1 \vee x_2 \vee \neg x_3)$		1	0	1	0	0	1	0	0	164
$\wedge\ \psi_2$:	$(\neg x_2 \vee x_3 \vee \neg x_4)$	$\Leftrightarrow$	0	0	0	1	1	0	0	1	25
$\wedge\ \psi_3$:	$(\neg x_1 \vee \neg x_3 \vee x_4)$		0	1	0	0	0	1	1	0	70

TABLE I

EXAMPLE OF MATRIX REPRESENTATION AND SIGNATURES OF A 3-CNF FORMULA.

There are $2^3 \times C_3^n$ possible clauses with n variables. A 3-CNF formula with *dimension* (k,m) with $k \leq n$ is composed of m different clauses drawn from the $2^3 \times C_3^n$ possible clauses. So, the total number of such formulae is

$$C_m^{2^3 \times C_3^n} = \frac{(2^3 \times C_3^n)!}{m! \times (2^3 \times C_3^n - m)!} = \mathcal{O}(n^{3m}) \tag{2}$$

Let $\Phi_{n,m}$ denote the set of all these formulae :

$$\Phi_{n,m} = \{\varphi : \varphi \text{ is a 3-CNF formula of } \textit{dimension } (k,m) \text{ with } k \leq n \ \}$$

The 3-CNF-Satisfiability problem is to find a function Ξ :

$$\begin{aligned} \Xi : \Phi_{n,m} &\longrightarrow \{0,1\} \\ \varphi \quad &\rightsquigarrow \quad 0 \text{ if } \varphi \text{ is non satisfiable and 1 otherwise} \end{aligned} \tag{3}$$

The 3-CNF-Satisfiability problem is known to belong to the $\boldsymbol{NP}$ class.

II. A "Meta Mathematical" proof that $\boldsymbol{P} \neq \boldsymbol{NP}$ is impossible to prove

One way to prove that $\boldsymbol{P} \neq \boldsymbol{NP}$ is to show that the complexity measure $T_M(n)$ for some $\boldsymbol{NP}$ problem, like the 3-CNF-SAT problem, cannot be reduced to a polynomial time. We will show that the 3-CNF-SAT problem behaves as a common *safe problem* and that its complexity is time dependent. In fact, at some specific time $t_0 + \Delta t$, the 3-CNF-SAT problem will be of polynomial complexity. So, $\boldsymbol{P} \neq \boldsymbol{NP}$ will not be provable, as $T_M(n)$ is not *"always"* supra-polynomial.

A. The analogy with the safe problem and the time dependent nature of complexity

Finding whether or not a given 3-CNF formula φ is satisfiable is like being in front of a *safe*, trying to find the *opening combination*. One has to try any possible value (0 or 1) for the variable x_i in φ to see whether some combination satisfies φ, in the same way as one tries any combination to get the one, if it exists, that opens the safe.

Let us consider more deeply the analogy between the 3-CNF-SAT problem and the *safe*

problem, especially by looking to the *time dependent* nature of the complexity involved here. It is clear that when you are in front of a safe for the first time, it is a very hard problem, as you do not have any information about the correct opening combination. In fact, in the worst case, it takes an exponential time to find it. But as soon as you have succeeded in opening the safe (or in finding that there is no solution), the problem becomes trivial. It takes only one operation to open the safe or to declare it impossible to open.

Let us denote by t_0 the first time you try to open the safe, and by Δt the time needed to find the solution. Let us remark that Δt can be huge but it is always finite as the number of possible combinations is finite. Now we compute the complexity measure $T_{safe}(n)$ for the *safe problem* at t_0 and $t_0 + \Delta t$.

In t_0, one has to test all possible combinations. If the safe has n buttons with only two positions (0 or 1), there will be 2^n possibilities. Because no information is available about the solution, there is no way to reduce the number of cases to be tested. The exponential complexity of the problem comes from the total lack of information about the solution. This absence of information is strictly related to *the random nature of the problem* : the finding of the opening combination is a random search process for anyone in front of the safe, at least in t_0. So, we get

$$T_{safe,\, t_0}(n) = 2^n$$

But after Δt, the correct opening combination is known *forever*, and the complexity measure is now

$$T_{safe,\, t_0+\Delta t}(n) = 1$$

As one can see, the complexity measure $T_{safe}(n)$ for the safe problem is *time dependent*.

The same occurs for the 3-CNF-SAT problem as well as for any ***NP*** problem. Their complexity measure changes in time. The idea of this section about the impossibility to prove ***P*** $\neq$ ***NP*** is to show that, even if $T_{3-CNF-SAT,\, t_0}(n)$ is not known (exponential or polynomial ?), there exists some Δt, even huge, such that the complexity measure is polynomial in $t_0 + \Delta t$.

B. The Computation of $T_{3-CNF-SAT,\, t_0+\Delta t}(n)$

Let us take Δt large enough so that Ξ [the 3-CNF-SAT decision function, see equation (3)] is known for all the 3-CNF formulae in $\Phi_{n,m}$. Δt exists and is finite. In the analogy with the safe problem, it corresponds to the time needed to find the solution for all safe equipments of dimension n. Until now, we do not know whether Ξ can be computed in polynomial time or not, but this only changes the size of Δt.

The output of Ξ is the set $\mathcal{S}_{n,m}$ of all satisfiable 3-CNF formulae of $\Phi_{n,m}$, or equivalently $\overline{\mathcal{S}}_{n,m} = \Phi_{n,m} \setminus \mathcal{S}_{n,m}$, the set of all non satisfiable 3-CNF formulae. As equation (2) shows, $\overline{\mathcal{S}}_{n,m}$ contains at most $\mathcal{O}(n^{3m})$ elements. The worst case occurs when $m = (2^3 \times C_3^n)/2 = \mathcal{O}(n^3)$. As $\overline{\mathcal{S}}_{n,m} \subseteq \Phi_{n,m}$, the equation (2) gives us the following result :

$$\#\{\overline{\mathcal{S}}_{n,m}\} < \#\{\Phi_{n,m}\} = \mathcal{O}(n^{3(n^3)}) \quad \Rightarrow \quad \#\{\overline{\mathcal{S}}_{n,m}\} = \mathcal{O}(2^{n^3}) \quad \text{as } n^3 > 2 \tag{4}$$

See Figure 1 for an example of $\#\{\Phi_{n,m}\}$ and $\#\{\overline{\mathcal{S}}_{n,m}\}$ with $n = 4$. The figure shows that $\#\{\Phi_{n,m}\}$ and $\#\{\overline{\mathcal{S}}_{n,m}\}$ behaves similarly.

So, one can now calculate $T_{3-CNF-SAT,\, t_0+\Delta t}(n)$: it is the time required to check whether a specific 3-CNF formula belongs or not in $\overline{\mathcal{S}}_{n,m}$, after Δt large enough for the entire set $\overline{\mathcal{S}}_{n,m}$ to be computed. If one can allocate an exponential space for memory to save the elements of $\overline{\mathcal{S}}_{n,m}$ (as accepted in Turing machines), then a hash algorithm, based on the *clause's signatures*, can be used to see whether a 3-CNF formula φ belongs or not to the set $\overline{\mathcal{S}}_{n,m}$. For instance, one can use u_i, the i^{th} *ordered signature of clauses*, as the i^{th} successive hash function $h_i(\varphi)$. It takes $\mathcal{O}(2n)$ operations to compute each of these m *clause's signatures* of φ and $\mathcal{O}(m \log m)$ computations to sort them. We need then $\mathcal{O}(2^3 \times C_3^n)$ operations, which corresponds to the maximum number of possible values for the signatures, to find whether the signature belongs or not to the corresponding section of $\overline{\mathcal{S}}_{n,m}$ where the formulae are also ordered, in a lexical ordering, following their clause's signatures. Using equation (1) [*i.e.* $\mathcal{O}(m) = \mathcal{O}(n^3)$],

$$\begin{aligned} T_{3-CNF-SAT,\, t_0+\Delta t}(n) &= \mathcal{O}(m(2n) + (m \log m) + m(2^3 C_3^n)) \\ &= \mathcal{O}(m^2) = \mathcal{O}(n^k) \quad \text{for some } k \in I\!N \end{aligned} \tag{5}$$

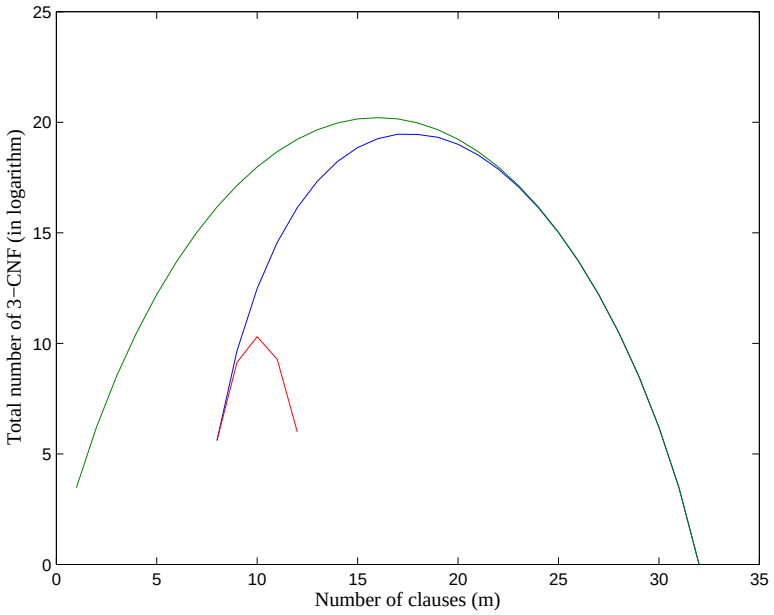

Fig. 1

LOGARITHMIC SCALE : THE UPPER CURVE REPRESENTS THE TOTAL NUMBER OF ALL POSSIBLE 3-CNF IN $\Phi_{4,m}$; THE SECOND ONE, THE TOTAL OF NON-SATISFIABLE 3-CNF, *i.e.* $\#\{\overline{\mathcal{S}}_{4,m}\}$, AND THE LOWER ONE, THE TOTAL OF *Irreducible Non-Satisfiable* 3-CNF, *i.e.* $\#\{\overline{\mathcal{S}}_{4,m}^{INS}\}$ (SEE SECTION B FOR THE DEFINITION).

C. The "unprovability" of $\boldsymbol{P \neq NP}$

Theorem II.1: It is impossible to prove that $\boldsymbol{P \neq NP}$ in the deterministic or time independent framework of Mathematics.

Proof: The solution of the 3-CNF-SAT problem is equivalent to the setting of these two functions Ξ' and Ξ" :

$$\begin{aligned} (\text{In } t_0) \quad \Xi' : \Phi_{n,m} &\xrightarrow{\mathcal{O}(?)} \{0,1\} \qquad \textit{(the construction of } \overline{\mathcal{S}}_{n,m}\textit{)} \\ \varphi \quad &\rightsquigarrow \quad 0 \quad \text{if } \varphi \in \overline{\mathcal{S}}_{n,m} \text{ and 1 otherwise} \qquad (6) \\ (\text{In } t_0 + \Delta t) \quad \Xi'' : \Phi_{n,m} &\xrightarrow{\mathcal{O}(n^k)} \{0,1\} \qquad \textit{(}\varphi \overset{?}{\in} \overline{\mathcal{S}}_{n,m} \textit{ when } \overline{\mathcal{S}}_{n,m} \textit{ is known)} \\ \varphi \quad &\rightsquigarrow \quad 0 \quad \text{if } \varphi \in \overline{\mathcal{S}}_{n,m} \text{ and 1 otherwise} \qquad (7) \end{aligned}$$

The *meta mathematical argument* lies in the fact that any operation done by Ξ' in t_0 can be reduced to a polynomial time operation by Ξ'' in $t_0 + \Delta t$ [1].

[1] To make it easier to understand, let us think of the version of 3-CNF-SAT with $n = 4$: it took us several months to build $\overline{\mathcal{S}}_{n,m}$, but now it only takes seconds to solve the 3-CNF-SAT problem with 4 variables. And this is done

Mathematically speaking, it is impossible to make a formal or mathematical distinction between both functions Ξ' and Ξ", as time does not interfere with proofs in mathematics. More precisely, if someone proves that the 3-CNF-SAT problem Ξ (or Ξ') is non polynomial, this assertion, as well as the steps for the demonstration, should be true at any time, independently of t, even in $t_0 + \Delta t$. The proof could not introduce time in the demonstration. But people will only be able to proof the non polynomial nature of 3-CNF-SAT for time t_0, certainly not for time $t_0 + \Delta t$ as shown in equation (5). And this argument holds for all $\boldsymbol{NP}$ problems because all of them are equivalent, in term of complexity, to the 3-CNF-SAT problem. ■

This is exactly the same situation as with the safe problem : the complexity measure of the problem is changing over time, becoming polynomial after some large Δt. But the $\boldsymbol{P}$ - $\boldsymbol{NP}$ question does not consider time as far as complexity is concerned : if we do not consider the time dependent nature of complexity, one should conclude that $\boldsymbol{P} = \boldsymbol{NP}$. The next section will show that it is not so clear.

III. A "Meta Mathematical" proof that $\boldsymbol{P} = \boldsymbol{NP}$ is impossible to prove

A. The "$\boldsymbol{P} = \boldsymbol{NP}$" assertion is not equivalent to "Not $\boldsymbol{P} \neq \boldsymbol{NP}$"

The previous *time dependent* argument is no longer valid with respect to $\boldsymbol{P}=\boldsymbol{NP}$, as we can have $T_{M,t_0}(n) = T_{M,t_0+\Delta t}(n) = \mathcal{O}(n^k)$ in this case. Indeed, from a strict mathematical point of view, one should accept that $\boldsymbol{P} = \boldsymbol{NP}$ as soon as $\boldsymbol{P} \neq \boldsymbol{NP}$ is proven to be impossible. But, if we take into account the time dependence of the complexity measure $T_M(n)$, the assertion "$\boldsymbol{P}=\boldsymbol{NP}$" does not mean solely the contrary of "$\boldsymbol{P}\neq\boldsymbol{NP}$", even if both assertions are mutually exclusive.

Indeed, "$\boldsymbol{P}=\boldsymbol{NP}$" can be rewritten as

$$T_{M,t}(n) = T_{M,t+\Delta t}(n) = \mathcal{O}(n^k) \quad \forall t, \Delta t \text{ and for any problem } M \text{ in } \boldsymbol{NP} \tag{8}$$

forever. A similar reasoning can be done for the i^{th} decimal of π, or for the list of the n first prime numbers.

The idea in this section is to show that for any $\boldsymbol{NP}$ problem M, there will be a time t where $T_{M,t}(n) >> \mathcal{O}(n^k)$ [$T_{M,t}(n) = \Omega(2^n)$, for instance[2]]. Therefore, equation (8) will not hold and the assertion "$\boldsymbol{P}=\boldsymbol{NP}$" will be false. The idea here is to point out some *random property* related to a special class of 3-CNF formulae, the *INS* 3-CNF formulae, as we did with the *safe problem* when we computed $T_{safe,t_0}(n)$.

B. The class of INS 3-CNF formulae

Let us first introduce the notion of *Irreducible Non Satisfiable* (or *INS*) 3-CNF formulae.

Definition III.1: An *INS 3-CNF formula* is a non satisfiable 3-CNF formula $\varphi^*_{n,m}$ such that any smaller sub-formulae $\varphi_{k,l}$ ($k \leq n$, $l \leq m$) of $\varphi^*_{n,m}$ is satisfiable. This means that the non satisfiability nature of $\varphi^*_{n,m}$ requires the entire set of the m clauses of $\varphi^*_{n,m}$.
The argument in the following section is to divide the 3-CNF-SAT problem into two separated and *"orthogonal"* problems : the *INS-3-CNF-SAT* and the *INS-Reduction* problems.

C. The "unprovability" of $\boldsymbol{P}=\boldsymbol{NP}$

Lemma III.1: For some time $t+\delta t$, the 3-CNF-SAT problem is $\Omega(2^n)$, even if one can solve the INS-3-CNF-SAT problem in $\mathcal{O}(n^k)$.

Proof: The *core of this proof* is to concentrate our attention, not on the satisfiability characteristic of $\varphi_{n,m}$, but on the non necessary clauses in $\varphi_{n,m}$.

1. Let us suppose that, for some time t, we have got enough time to build the set $\overline{\mathcal{S}}^{INS}_{n,m}$ of all the *INS* 3-CNF $\varphi^*_{n,m}$. As shown in equation (5), at time t, it takes $\mathcal{O}(n^k)$ computations to check whether or not a given formula $\varphi^*_{n,m}$ belongs to $\overline{\mathcal{S}}^{INS}_{n,m}$, as $\overline{\mathcal{S}}^{INS}_{n,m} \subseteq \overline{\mathcal{S}}_{n,m}$.

2. Let $\varphi^*_{n,m}$ be an *INS* 3-CNF formula in $\overline{\mathcal{S}}^{INS}_{n,m}$. From $\varphi^*_{n,m}$, we generate a new non satisfiable formula $\varphi_{n,2m}$ with $2m$ clauses, by adding randomly m extra clauses. These clauses can be considered as noisy extra clauses. This random generation is over at time $t+\delta t$.

3. At time $t+\delta t$ (remember that we have knowledge of $\overline{\mathcal{S}}^{INS}_{n,m}$, from time t), we want to check whether or not $\varphi_{n,2m}$ belongs to $\overline{\mathcal{S}}_{n,2m}$ [the general 3-CNF-SAT problem, with no information about $\overline{\mathcal{S}}_{n,2m}$].

[2] $\Omega(2^n)$ means that the computation time is larger than 2^n (*i.e.* exponential).

Our 3-CNF-SAT algorithm on $\varphi_{n,2m}$ will use the information about $\overline{\mathcal{S}}_{n,m}^{INS}$ as this information is related to the most difficult part of the algorithm (the non satisfiability property of a 3-CNF formula). Moreover, by hypothesis, at time $t+\delta t$, this sub-algorithm is supposed to be polynomial for any *INS* 3-CNF formula.

So, the 3-CNF-SAT algorithm will have to find, inside the clauses of $\varphi_{n,2m}$, the added or noisy clauses, so that it can find the hidden *INS* sub-formula $\varphi_{n,m}^*$ in $\varphi_{n,2m}$. Let us call this search *the INS-Reduction* problem. We have thus divided the 3-CNF-SAT problem in two orthogonal problems : the *INS-3-CNF-SAT* problem (in $\mathcal{O}(n^k)$) and the *INS-Reduction* problem.

4. Let us now prove that, at time $t+\delta t$, the *INS-Reduction* problem is $\Omega(2^n)$ for our 3-CNF formula $\varphi_{n,2m}$. Once again, we use a *meta mathematical* argument, based on some property of true randomness.

To the Irreducible Non Satisfiable formula $\varphi_{n,m}^*$, one can add any extra clause without changing the non satisfiable nature of the obtained formula. These added clauses can be selected in a totally arbitrary way, with respect to $\varphi_{n,m}^*$ (except that all clauses should be unique). So, one can add to $\varphi_{n,m}^*$ many different clauses, *in a random way*, without link with $\varphi_{n,m}^*$. One possible random output of this generation process can be our peculiar formula $\varphi_{n,2m}$.

In fact, $\varphi_{n,2m}$ can be seen as the final output at time $t+\delta t$ of a random process beginning with $\varphi_{n,m}^*$ at time t. If we look at the process in a backward way, we see that there are C_m^{2m} different possible random processes beginning with different $\varphi_{n,m}^*$, which lead to a peculiar $\varphi_{n,2m}$. Mathematically speaking, it is *impossible to distinguish the given formula $\varphi_{n,2m}$ from the result of a true random process.* And if $\varphi_{n,2m}$ is truly a random output, we are then in presence of a problem similar to the *safe problem* in time t_0, when a random search process was needed to find the solution. There is no way to get useful information for the search of $\varphi_{n,m}^*$ inside $\varphi_{n,2m}$ [the *INS-Reduction* problem]. So, one has to check all possible combinations for the sub-formula $\varphi_{n,m}^*$ and then see whether this sub-formula belongs or not to $\overline{\mathcal{S}}_{n,m}^{INS}$ (in $\mathcal{O}(n^k)$).

And this *INS-Reduction* algorithm takes at least an exponential number of operations

(in fact[3] : $C_m^{2m} = \Omega(2^m)$). Using the fact that $m = \Omega(n)$ (see equation (1)), the 3-CNF-SAT problem for any formula like $\varphi_{n,2m}$ is $\Omega(2^m \times n^k) = \Omega(2^m) = \Omega(2^n)$, at least at time $t + \delta t$.

■

Theorem III.2: Any 3-CNF-SAT algorithm should contain, at some time $t + \delta t$, a sub-algorithm equivalent to the *INS-Reduction* algorithm, and therefore is $\Omega(2^n)$. So, it is impossible to prove that $\boldsymbol{P} = \boldsymbol{NP}$, in the sense defined in equation (8).

Proof: The proof of this assertion is based on the very nature of the $2m$ clauses of $\varphi_{n,2m}$: m of them are mathematically related to the non satisfiability property of $\varphi_{n,2m}$, while the other m clauses are totally unrelated (as noise) to it. Any 3-CNF-SAT algorithm for such formula as $\varphi_{n,2m}$ should handle, in some way, these noisy extra clauses. And, as these extra clauses can be anything (totally random), there is no way to escape some exponential *INS-Reduction* (or random search) process to get rid of them. ■

Once again, the *pseudo random nature* of the $\boldsymbol{NP}$ problem arises in the reflection. It is because of the possible randomness within the generation of the extra clauses (from $\varphi^*_{n,m}$ to $\varphi_{n,2m}$) that there is no efficient or polynomial way to find back $\varphi^*_{n,m}$ inside $\varphi_{n,2m}$, and thus 3-CNF-SAT cannot be proved to be in $\boldsymbol{P}$ because of that.

IV. Conclusions

This paper tries to show that the $\boldsymbol{P} \stackrel{?}{=} \boldsymbol{NP}$ problem is impossible to solve within the time independent framework of Mathematics, as neither $\boldsymbol{P} = \boldsymbol{NP}$ nor $\boldsymbol{P} \neq \boldsymbol{NP}$ can be proved without reference to time. The key concept of the paper is the temporal nature of the complexity measure for the $\boldsymbol{NP}-$hard problems. This time dependence is closely related to some (pseudo) randomness in the heart of these problems. Some analogy can be found with the Chaos theory, when pseudo randomness arises from deterministic processes.

For the author, $\boldsymbol{NP}$ is really different from $\boldsymbol{P}$ but the difference lies in the distinction between true randomness and mathematical pseudo-randomness, and this frontier is situated

[3] See Appendix for a proof.

on the limit border of Mathematics (which is deterministic).

The impossibility for a solution to $\boldsymbol{P} \stackrel{?}{=} \boldsymbol{NP}$ gives a new perspective on the *pseudo non deterministic (or random)* nature of the most difficult problems, the $\boldsymbol{NP}-$hard problems : we can see these problems as so inextricable that we are in front of them like someone facing some random search problem (as the safe problem), even if they are deterministic (not random) in their very essential nature, *i.e.* as quasi chaotic problems.

Therefore, the $\boldsymbol{P}-\boldsymbol{NP}$ *"unprovability"* can be seen as the expression of the incapacity for Mathematics to give a time independent definition of randomness.

V. Appendix : Details about the exponential complexity of the INS-Reduction Process

A. Preliminaries

Let $\varphi^*_{n,2m}$ be the 3-CNF formula to be reduced, and $\varphi_{n,p}$ be any sub-formulae of $\varphi^*_{n,2m}$. We suppose that $\varphi^*_{n,2m}$ is a random extension of some $\varphi_{n,m}$ in $\overline{\mathcal{S}}^{INS}_{n,m}$, where $\overline{\mathcal{S}}^{INS}_{n,p}$ denotes the set of all *Irreducible Non Satisfiable 3-CNF* formulae $\varphi_{n,p}$ of *dimension* (n,p). These sets are supposed to be known here.

The *INS-Reduction Process* checks whether there exists $\varphi_{n,p}$ in $\overline{\mathcal{S}}^{INS}_{n,p}$, for some $p \leq 2m$, such that $\varphi_{n,p}$ is a sub-formula of $\varphi^*_{n,2m}$ and $\varphi_{n,p}$ is *Irreducible Non Satisfiable*. We will prove that this process has an exponential complexity : $T_{INS-Reduction,t+\delta t}(n) = \Omega(2^n)$.

B. The two approaches for the INS-Reduction Process

In fact, there are only two major ways to check whether or not there exists a sub-formula of $\varphi^*_{n,2m}$ in $\overline{\mathcal{S}}^{INS}_{n,p}(p < 2m)$. Any *INS-Reduction* algorithm will be a mixture of these two approaches :

1. From $\varphi^*_{n,2m}$ to $\overline{\mathcal{S}}^{INS}_{n,p}$: one considers all the possible sub-formulae of $\varphi^*_{n,2m}$

with *dimension* (n,p) $(p<2m)$, and then checks whether these sub-formulae belong to $\overline{\mathcal{S}}_{n,p}^{INS}$; one stops as soon as such a sub-formula is found. By hypothesis, the algorithm will stop with $p=m$ as $\varphi_{n,2m}^{*}$ is an extension of some $\varphi_{n,m}\in\overline{\mathcal{S}}_{n,m}^{INS}$.

2. From $\overline{\mathcal{S}}_{n,p}^{INS}$ to $\varphi_{n,2m}^{*}$: for each formula $\varphi_{n,p}$ in $\overline{\mathcal{S}}_{n,p}^{INS}(p<2m)$, one checks whether $\varphi_{n,p}$ is a sub-formula of $\varphi_{n,2m}^{*}$; one stops as soon as such a sub-formula is found. Here again, $p=m$ at the end of the process.

C. Complexity of both approaches

1. Because of the *pseudo random* nature of $\varphi_{n,2m}^{*}$, the first algorithm is required to consider all the sub-formulae of $\varphi_{n,2m}^{*}$ of *dimension* $(n,p)(p<2m)$. As $\varphi_{n,2m}^{*}$ is an extension of some $\varphi_{n,m}$, the algorithm will consider $\sum_{p=1}^{m}C_{p}^{2m}=\Omega(2^m)$ different sub-formulae. For each of these sub-formulae, it takes $\mathcal{O}(n^k)$ operations (see equation (5)) to check whether or not it belongs to $\overline{\mathcal{S}}_{n,p}^{INS}$. So, the first algorithm is $\Omega(2^m)\times\mathcal{O}(n^k)=\Omega(2^m)=\Omega(2^n)$.

2. Because of the *pseudo random* nature of $\varphi_{n,2m}^{*}$, the second algorithm is required to consider all the formulae belonging to $\overline{\mathcal{S}}_{n,p}^{INS}(p<2m)$. As $\varphi_{n,2m}^{*}$ is an extension of some $\varphi_{n,m}$, the algorithm will consider $\sum_{p=1}^{m}\#\{\overline{\mathcal{S}}_{n,p}^{INS}\}$ different *INS 3-CNF formulae*. For each of these formulae, it takes $\mathcal{O}(n^k)$ operations to check whether one gets or not a sub-formula of $\varphi_{n,2m}^{*}$. This is just a classical *string searching algorithm*, which has polynomial complexity. So, the complexity of the second algorithm will be $\sum_{p=1}^{m}\#\{\overline{\mathcal{S}}_{n,p}^{INS}\}\times\mathcal{O}(n^k)$.

By proving in the next section that $\sum_{p=1}^{m}\#\{\overline{\mathcal{S}}_{n,p}^{INS}\}$ is $\Omega(2^n)$, we show that both approaches for the *INS-Reduction* process are equivalent in terms of complexity. And this holds for any mixture of these approaches.

D. Theorem : $\displaystyle\sum_{p=1}^{m}\#\{\overline{\mathcal{S}}_{n,p}^{INS}\}=\Omega(2^n)$ *for* $m\geq\frac{C_3^n\,2^n}{2^{n-3}+C_3^n-1}$

D.1 Notations

Let $\varphi_{n,m}\in\Phi_{n,m}$ be a 3-CNF formula with propositional variables $x_1,\cdots,x_n$ and clauses $\psi_1,\cdots,\psi_m$. Let Ψ_n be the set of the $2^3\times C_3^n$ possible clauses with n variables, and $\{0,1\}^n$ be the set of all possible logical values for the variables.

Let $g:\Psi_n \rightarrow \{0,1\}^n : \psi_j \rightsquigarrow g(\psi_j) = \mathcal{S}_j \subseteq \{0,1\}^n$ where $\mathcal{S}_j = \{v \in \{0,1\}^n : \psi_j(v) = 0\}$. For instance, $\psi_j = x_1 \vee x_2 \vee x_4 (\in \Psi_4)$ leads to $g(\psi_j) = \mathcal{S}_j = \{(0,0,0,0),(0,0,1,0)\}$. See `Table II` where $v = (a,b,c,d)$ corresponds to the column $i = a + b.2 + c.2^2 + d.2^3$. It is clear that $\#\mathcal{S}_j = 2^{n-3}$.

Let us define $g^{\Leftarrow}(v|\varphi_{n,m}) = \{\psi_j : \psi_j(v) = 0, \psi_j \text{ in } \varphi_{n,m}\}$. We have that $\#\{g^{\Leftarrow}(v|\varphi_{n,m})\} \leq C_3^n$ as each v can correspond to maximum C_3^n clauses.

ψ_j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\neg x_1 \vee \neg x_2 \vee \neg x_3$	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
$x_1 \vee \neg x_2 \vee \neg x_3$	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
$\boxed{\neg x_1 \vee x_2 \vee \neg x_3}$	0	0	0	0	0	$\boxed{1}$	0	0	0	0	0	0	0	$\underline{1}$	0	0
$x_1 \vee x_2 \vee \neg x_3$	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
$\neg x_1 \vee \neg x_2 \vee x_3$	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
$x_1 \vee \neg x_2 \vee x_3$	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
$\neg x_1 \vee x_2 \vee x_3$	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
$x_1 \vee x_2 \vee x_3$	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
$\neg x_1 \vee \neg x_2 \vee \neg x_4$	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
$x_1 \vee \neg x_2 \vee \neg x_4$	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
$\neg x_1 \vee x_2 \vee \neg x_4$	0	0	0	0	0	0	0	0	0	1	0	0	0	$\underline{1}$	0	0
$x_1 \vee x_2 \vee \neg x_4$	0	0	0	0	0	0	0	0	1	0	0	0	1	$\underline{0}$	0	0
$\neg x_1 \vee \neg x_2 \vee x_4$	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
$x_1 \vee \neg x_2 \vee x_4$	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
$\neg x_1 \vee x_2 \vee x_4$	0	$\underline{1}$	0	0	0	$\underline{1}$	0	0	0	0	0	0	0	0	0	0
$x_1 \vee x_2 \vee x_4$	1	$\underline{0}$	0	0	1	$\underline{0}$	0	0	0	0	0	0	0	0	0	0
$\neg x_1 \vee \neg x_3 \vee \neg x_4$	0	0	0	0	0	0	0	0	0	0	0	0	0	$\underline{1}$	0	1
$x_1 \vee \neg x_3 \vee \neg x_4$	0	0	0	0	0	0	0	0	0	0	0	0	1	$\underline{0}$	1	0
$\neg x_1 \vee x_3 \vee \neg x_4$	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
$x_1 \vee x_3 \vee \neg x_4$	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
$\neg x_1 \vee \neg x_3 \vee x_4$	0	0	0	0	0	$\underline{1}$	0	$\underline{1}$	0	0	0	0	0	0	0	0
$x_1 \vee \neg x_3 \vee x_4$	0	0	0	0	1	$\underline{0}$	1	$\underline{0}$	0	0	0	0	0	0	0	0
$\neg x_1 \vee x_3 \vee x_4$	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
$x_1 \vee x_3 \vee x_4$	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
$\neg x_2 \vee \neg x_3 \vee \neg x_4$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
$x_2 \vee \neg x_3 \vee \neg x_4$	0	0	0	0	0	0	0	0	0	0	0	0	1	$\underline{1}$	0	0
$\neg x_2 \vee x_3 \vee \neg x_4$	0	0	0	0	0	0	0	0	0	0	1	1	0	$\underline{0}$	0	0
$x_2 \vee x_3 \vee \neg x_4$	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
$\neg x_2 \vee \neg x_3 \vee x_4$	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
$x_2 \vee \neg x_3 \vee x_4$	0	0	0	0	$\underline{1}$	$\underline{1}$	0	0	0	0	0	0	0	0	0	0
$\neg x_2 \vee x_3 \vee x_4$	0	0	1	1	$\underline{0}$	$\underline{0}$	0	0	0	0	0	0	0	0	0	0
$x_2 \vee x_3 \vee x_4$	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TABLE II

GRAPH OF g : *Value* 1 *in the* i^{th} *column corresponds to* $v \in \mathcal{S}_j$, *where* $v = (a,b,c,d)$ *with* $a + b.2 + c.2^2 + d.2^3 = i$. *The box is the first pivot* v_1 *and the underlined elements are the discarded values for the next pivot* v_2 *(see section (D.3)).*

D.2 Sufficient and necessary conditions for non satisfiability

Theorem V.1: A 3-CNF formula $\varphi_{n,m} = \bigwedge_{j=1}^{m} \psi_j$ is not satisfiable iff $\bigcup_{j=1}^{m} g(\psi_j) = \bigcup_{j=1}^{m} \mathcal{S}_j = \{0,1\}^n$.

Proof:

$$\begin{aligned}
\varphi_{n,m} \text{ is not satisfiable } &\Leftrightarrow \forall v \in \{0,1\}^n \ : \ \varphi_{n,m}(v) = 0 \\
&\Leftrightarrow \forall v \in \{0,1\}^n \ : \ \exists j : \psi_j(v) = 0 \\
&\Leftrightarrow \forall v \in \{0,1\}^n \ : \ \exists j : v \in \mathcal{S}_j \\
&\Leftrightarrow \{0,1\}^n = \bigcup_{j=1}^{m} \mathcal{S}_j
\end{aligned}$$

■

Corollary V.1: As $\mathcal{S}_j \subseteq \{0,1\}^n$, a 3-CNF formula $\varphi_{n,m}$ is not satisfiable iff $\#\{\bigcup_{j=1}^{m} \mathcal{S}_j\} = 2^n$.

Looking at Table II, it is easy to verify if some formula $\varphi_{4,m}$ is satisfiable or not : one just has to check the existence of a "1" in each column, when one looks only at the lines corresponding to the clauses ψ_j of $\varphi_{4,m}$. For instance, it is clear that the first 8 clauses taken together are non satisfiable.

Theorem V.2: A 3-CNF formula $\varphi_{n,m}$ is not satisfiable in the INS-3-CNF-SAT sense, i.e. $\varphi_{n,m} \in \{\overline{\mathcal{S}}_{n,m}^{INS}\}$, iff $\bigcup_j \mathcal{S}_j = \{0,1\}^n$ and $\forall\, \psi_j, \exists\, v \in \mathcal{S}_j$ such that $g^{\Leftarrow}(v|\varphi_{n,m}) = \{\psi_j\}$.

Definition V.3: This variable v is called the *pivot* for ψ_j.

Proof: On the contrary, let us suppose that $\bigcup_j \mathcal{S}_j = \{0,1\}^n$ but $\exists\, \psi_j$ such that $\forall\, v \in \mathcal{S}_j, g^{\Leftarrow}(v|\varphi_{n,m}) \neq \{\psi_j\}$. Of course, $\psi_j \subseteq g^{\Leftarrow}(v|\varphi_{n,m})$ as $v \in \mathcal{S}_j$. Thus,

$$\begin{aligned}
&\forall\, v \in \mathcal{S}_j, \exists\, \psi_k^v \neq \psi_j \text{ such that } \psi_k^v \subseteq g^{\Leftarrow}(v|\varphi_{n,m}) \\
\Leftrightarrow\ &\forall\, v \in \mathcal{S}_j, \exists\, k \neq j \text{ such that } v \in \mathcal{S}_k \\
\Leftrightarrow\ &\bigcup_{i=1}^{m} \mathcal{S}_i = \bigcup_{\substack{i=1 \\ i \neq j}}^{m} \mathcal{S}_i = \{0,1\}^n \\
\Leftrightarrow\ &\varphi_{n,m} \text{ is not satisfiable, even when the clause } \psi_j \text{ is deleted.} \\
\Leftrightarrow\ &\varphi_{n,m} \text{ is not an Irreducible Non Satisfiable formula.}
\end{aligned}$$

■

From Table II, we see that a formula $\varphi_{4,m}$ is a INS formula if for each clause in $\varphi_{4,m}$ there exists at least one column with only one "1" in it. For instance, it is clear that the first 9 clauses taken together are not *Irreducible Non Satisfiable*, as there exists two "1" in columns

11 and 15 for the 9^{th} clause. This last clause can be considered as a noisy clause, as there is no *pivot* for it.

Theorem V.4: Only formulae $\varphi_{n,m}$ with $8 \leq m \leq \frac{C_3^n\, 2^n}{2^{n-3}+(C_3^n-1)} \stackrel{def}{=} m_{max}$ can be non satisfiable in the INS-3-CNF-SAT sense.

Proof: • As $\#\mathcal{S}_j = 2^{n-3}$:

$$\begin{aligned}
\varphi_{n,m} \text{ is non satisfiable } &\Leftrightarrow \bigcup_{j=1}^{m} \mathcal{S}_j = \{0,1\}^n \\
&\Rightarrow \sum_{j=1}^{m} \#\mathcal{S}_j \geq 2^n = \#\{0,1\}^n \\
&\Rightarrow m\, 2^{n-3} \geq 2^n \\
&\Rightarrow m \geq 8
\end{aligned}$$

• As $\varphi_{n,m}$ is not satisfiable in the INS-3-CNF-SAT sense :

$$\begin{aligned}
\varphi_{n,m} \text{ is not satisfiable } &\Leftrightarrow \bigcup_{j} \mathcal{S}_j = \{0,1\}^n \text{ and } \forall j\ \exists v_j \in \mathcal{S}_j \ :\ g^{\Leftarrow}(v_j) = \{\psi_j\} \\
&\Rightarrow \forall v \in \{0,1\}^n \ : \begin{cases} \#\{g^{\Leftarrow}(v|\varphi_{n,m})\} = 1 & \text{if } v = v_j \\ \#\{g^{\Leftarrow}(v|\varphi_{n,m})\} \leq C_3^n & \text{otherwise} \end{cases} \\
&\Rightarrow \sum_{v\in\{0,1\}^n} \#\{g^{\Leftarrow}(v|\varphi_{n,m})\} \leq m + C_3^n(2^n - m) \\
&\quad [\ \sum_{v\in\{0,1\}^n} \#\{g^{\Leftarrow}(v|\varphi_{n,m})\} \text{ is the total number} \\
&\quad \text{of relations between the } m \text{ clauses } \psi_j \text{ and } \{0,1\}^n\ ]
\end{aligned}$$

But we know that this total number of relations is also equal to

$$\sum_{j=1}^{m} \#\{\mathcal{S}_j\} = \sum_{j=1}^{m} 2^{n-3} = m \times 2^{n-3}$$

So, we have :

$$\begin{aligned}
m \times 2^{n-3} &\leq m + C_3^n(2^n - m) \\
m &\leq \frac{C_3^n\, 2^n}{2^{n-3} + C_3^n - 1} = m_{max}
\end{aligned}$$

■

D.3 An asymptotic exponential lower bound for $\sum_{p=1}^{m} \#\{\overline{\mathcal{S}}_{n,p}^{INS}\}$ for $m \geq m_{max}$

Theorem V.5:

$$\sum_{p=1}^{m} \#\{\overline{\mathcal{S}}_{n,p}^{INS}\} \geq \frac{\prod_{i=0}^{4}[2^n C_3^n - i\{2^{n-3}C_3^n + (2^{n-3}-1)(c_3^n-1)\}]}{\prod_{i=0}^{4}[\frac{C_3^n\, 2^n}{2^{n-3}+C_3^n-1} - i]}$$
$$= \Omega(2^n)$$

(for $m \geq m_{max} = \frac{C_3^n\, 2^n}{2^{n-3}+C_3^n-1}$)

Proof: The idea is to build the set of all INS-3-CNF formulae from the graph of g, by choosing recursively the *pivots* for these formulae. Indeed, any INS-3-CNF formula $\varphi_{n,m}$ is characterized by $(v_1, \cdots, v_m)$, where v_j is the *pivot* for the clause ψ_j. We will get $\sum \#\{\overline{\mathcal{S}}_{n,p}^{INS}\}$ by counting the number of possible choices for $(v_1, \cdots, v_m)$, with $8 \leq m \leq m_{max}$. Remember that $\forall v_j : g^{\Leftarrow}(v_j|\varphi_{n,m}) = \{\psi_j\}$ (see definition V.3).

• The first step is to choose a clause ψ_1 from the $2^3 C_3^n$ possible clauses, and then a *pivot* v_1 among the 2^{n-3} elements associated to ψ_1 (see `Table II` for an example of *pivot*). So, there is $2^n C_3^n$ possible choices for the first *pivot*.

• For the choice of the second clause ψ_2 and *pivot* v_2, we have to discard those clauses ψ such that $\psi \in g^{\Leftarrow}(v_1|\varphi_{n,1} = \psi_1)$, i.e. those clauses with a "1" in the column of v_1 in the table. C_3^n clauses (and the 2^{n-3} corresponding table elements) should be discarded at that stage, otherwise $g^{\Leftarrow}(v_1|\varphi_{n,2}) \neq \{\psi_1\}$. Looking at the other elements v in $\mathcal{S}_1$ (the line corresponding to ψ_1), we have to reject as future candidate for the next pivot, the elements of the table in the columns of these $v \in \mathcal{S}_1$. We should discard $(2^{n-3}-1)(C_3^n-1)$ elements, i.e. the number of elements in $\mathcal{S}_1$ different from v_1 times the number of non null elements in each column, not in $\mathcal{S}_1$. Indeed, if we take such an element as our next *pivot* v_2, we will get $g^{\Leftarrow}(v_2|\varphi_{n,2} = \psi_1 \wedge \psi_1) = \{\psi_1, \psi_2\} \neq \{\psi_2\}$ and that is contrary to the definition of a *pivot*. In `Table II`, this corresponds to the 3 underlined "1" in column 13, the third line $\mathcal{S}_1$ not being taken into account. In summary, there are $2^n C_3^n - \{2^{n-3}C_3^n + (2^{n-3}-1)(C_3^n-1)\}$ possibilities for the second *pivot* v_2. In `Table II`, this means "32 - 11" possibilities.

• The third (v_3) and following steps are similar. We discard the same number of elements from the table at each stage, *or less* if we choose a *pivot* such that some redundancy appears with previous deletions. From any previous choice of *pivots* $(v_1, v_2, v_3, \cdots)$, it is always possible to build at least one INS-3-CNF formula $\varphi_{n,m}$ with $8 \leq m \leq m_{max}$: as soon as there is no possible choice for the next *pivot*, this means that we have a INS-3-CNF formula. We then put the value "1" for the following terms in our product, so that the total number of possibilities remains unchanged. This is done by introducing the following term $\max\{1, [2^n C_3^n - i(2^{n-3}C_3^n + (2^{n-3}-1)(c_3^n - 1))]\}$ in our overall product.

• Let us note that for large n, the term $[2^n C_3^n - i\{2^{n-3}C_3^n + (2^{n-3}-1)(c_3^n-1)\}]$, which corresponds to the minimum value for the number of possibilities at stage i, becomes negative for $i \geq 5$, so we get that $\max\{1, [2^n C_3^n - i(2^{n-3}C_3^n + (2^{n-3}-1)(c_3^n - 1))]\} = 1 \; \forall \; i \geq 5$. We can thus limit our product to $i = 4$. Indeed, the overall product

$$\prod_{i=0}^{m_{max}-1} \max\{1, [2^n C_3^n - i(2^{n-3}C_3^n + (2^{n-3}-1)(c_3^n - 1))]\}$$
$$= \prod_{i=0}^{4} [2^n C_3^n - i\{2^{n-3}C_3^n + (2^{n-3}-1)(c_3^n-1)\}]$$

corresponds to a minimum value for the number of possible ways for choosing the five first *pivots* in the building of our INS-3-CNF formulae.

• We have now the *pivots* v_i and their corresponding clauses ψ_i, such that $g^{\Leftarrow}(v_i|\varphi_{n,m}) = \{\psi_i\}$. The number m of the so-selected *pivots* will depend on the selection, with $8 \leq m \leq m_{max}$. For each choice of $(v_0, \cdots, v_4)$, one can build a INS-3-CNF formula in $(m-5)!$ ways, depending on the ordering of $(v_5, \cdots, v_m)$. So, we get that $\prod_{i=0}^{4}[2^n C_3^n - i\{2^{n-3}C_3^n + (2^{n-3}-1)(c_3^n-1)\}] \times (m-5)!$ is a minimum value for the number of possible choices for the m *pivots*. Let us remark that these m *pivots*, as well as their corresponding m clauses can be selected in $m!$ different orders, as all these ordered selections are equivalent in terms of Irreducible Non Satisfiability. We should therefore retrieve the ordering by dividing by $m!$.

• Putting all together, we have

$$\prod_{i=0}^{4}[2^n C_3^n - i\{2^{n-3}C_3^n + (2^{n-3}-1)(c_3^n-1)\}] \times \frac{(m-5)!}{m!}$$

possible different INS-3-CNF formulae built with our m *pivots*. As the term $(m-5)!/m!$ depends on the selected *pivots*, we replace it by a lower bound :

$$\frac{(m-5)!}{m!} \geq \frac{(m_{max}-5)!}{(m_{max})!} = \frac{1}{\prod_{i=0}^{4}(m_{max}-i)}$$

Finally, we get :

$$\begin{aligned}
\sum_{p=1}^{m}\#\{\overline{\mathcal{S}}_{n,p}^{INS}\} &= \sum_{p=8}^{m_{max}}\#\{\overline{\mathcal{S}}_{n,p}^{INS}\} \\
&\geq \frac{\prod_{i=0}^{4}[2^n C_3^m - i\{2^{n-3}C_3^m + (2^{n-3}-1)(c_3^n-1)\}]}{\prod_{i=0}^{4}[m_{max}-i]} \\
&\geq \Omega(2^{5\,n}) \\
&= \Omega(2^n) \quad \text{for } m \geq m_{max} = \frac{C_3^n\, 2^n}{2^{n-3}+C_3^n-1}
\end{aligned}$$

■

References

[1] S. Cook. The P versus NP Problem. *Manuscript prepared for the Clay Mathematics Institute for the Millennium Prize Problems*, `http://www.claymath.org/millennium/ P_vs_NP/pvsnp.pdf`, November 2000.

[2] Th. Cormen, Ch. Leiserson, R. Rivest, and Cl. Stein. *Introduction to Algoritmics*. MIT Press, Cambridge, 2nd edition, 2001.

[3] A. Sanjeev and B. Boaz. *Computational Complexity : A Modern Approach, see* `http://www.cs.princeton.edu/theory/complexity/`. Cambridge University Press, Cambridge, to appear in 2009.

[4] M. Sipser. The History and Status of the P versus NP Question. *Proceedings of the 24th Annual Meeting ACM*, pages 603–618, 1992.