A proof that the set of NP-problems is bigger than the set of P-problems by using a logical consideration

AKRAM LOUIZ
Informatics
Landshut Hochschule, Germany
Coopérative Essalam, Rue Kacem Amine, N°10, Settat
MOROCCO

Abstract: - The field of informatics is the domain that emerged by applying the mathematical logic on electronic devices called computers in order to simplify many tasks for humans. The application of informatics in all economic and scientific areas is the most important factor that made our civilization reach our current phase of development. Nowadays, the experts and even the beginners of informatics are eager to use quantum computers. However, there is still an unsolved problem of classical theoretical informatics in ordinary electronic computers. It is the famous philosophical problem of the "Millenium Prize" of the Clay Mathematics Institute concerning the complexity of problems that has been treated by many other researchers but without acceptable sufficient answers. A solution to this problem can make all the fields based on informatics make huge progress. And thus, thanks to my short studies about informatics, I present to you this mathematical proof that deals with the sets of ***P*** problems, **NP-Complete** problems and **NP-Hard** problems in the field of classical electronic computers in order to prove new formulas about the cardinals of each group of complexity problems and about the intersections of each one of these sets. The aim is to contribute to an acceptable solution for this Millenium Problem and the methodology is purely logical and mathematical. The readers won't need any complicated notions from the background of previous informatics or mathematics research in order to understand the demonstrations of this article since the proof is based only on notions of sets by starting with easy logical considerations. Furthermore, you will find in this work a proof of an interesting theorem about the complexity of problems that allows us to identify **NP**-problems even if their algorithms have infinite time of execution. This paper ends by proving that the set of **NP**-Problems is definitely bigger than the set of **P**-Problems. Hence, all the readers are invited to understand and develop this work by inspecting the applied logical considerations in order to succeed in finding a sufficient solution to the interesting Millenium Problem of complexity.

Key-Words: - P problems ; NP problems ; NP-Complete problems ; NP-Hard problems ; complexity ; cardinality ; set ; execution time; Clay Mathematics; Millenium problem.

Received: June 9, 2022. Revised: July 25, 2023. Accepted: September 8, 2023. Published: October 3, 2023.

1 Introduction

I've been motivated by my short studies about informatics in order to produce a scientific article that treats informatics with easy mathematical logic. I've also been motivated by my previous work about the Landau-Siegel Zeros [1] that may contribute to a solution of the Millenium Prize of Riemann Hypothesis. Hence, maybe this work can be accepted as a contribution for the solution of the important Millenium problem about the complexity of problems.

This work concerns the cardinality and the intersections of the ***P***-problems, the NP problems and the NP-Hard problems which are presented in many works dealing with the notions of informatics and logic [2,3,4]. However, we will focus in this article on problems complexity by using the ordinary notions available when we deal with ordinary computers and not with nondeterministic Turing machines (NTM) [5,6,7] or Quantum Computers like it is explained in many works [8,9,10].

Most of the results of this work are made by considering that all NP-Hard problems that are not NP-Complete problems have higher complexity than NP-Complete problems and by considering that all NP-Complete problems have one same complexity that is directly linked to the value of a number M that should exist and which is a fixed number of ***P***-Problems. We also supposed the existence of a number of inputs that keeps all the P-Problems with finite times of execution but makes all NP-Hard problems with infinite times of execution.

2 The considered modelling

Let's consider that each decidable problem can be modelled by an algorithm that allows us to give an estimation of the execution time of that algorithm.

The execution time of the algorithms of ***P*** problems is finite. However, the execution times of NP-Hard problems and ***NP***-Complete problems are considered infinite since these problems have never been completely solved.

The set of ***P***-problems is composed of elements p_i with $i \in \mathbb{N}$ and i>0which are simple problems that have a finite time of execution.

All the set of ***decidable problems*** respects a law

where ($\times \Leftrightarrow$ And) is a product that makes the sum of the complexities of the problems of this product.

Consequently p_3=$p_1 \times p_2$means that the execution of the algorithm of the problem $p_3$is equivalent to the execution of the algorithm of the problem $p_1$followed by the execution of the algorithm of the problem p_2. Since the execution time of the algorithm of the problem $p_3$is finite, then $p_3$is also an element of the set of ***P***-problems. Consequently, an element of the set of ***P***-problems can be a product of ***n*** elements p_i with ***n*** is a finite natural number.

Let's consider that there is an element ***e*** of the set of ***P*** problems that has a null time of execution.

And thus:

$$e \times p_i = p_i \times e = p_i. \quad (1)$$

And we don't care in this work about the presence of a memory in the used computer. Hence, our computer can repeat executing the same element p_i even if it is a solved problem.

And thus, we consider that we have:

$$p_i \times p_i = p_i^2 \neq p_i \quad (2)$$

Hence, we consider that $p_i^n$is in the set of ***P*** problems for any element $p_i$and for any strictly positive natural number ***n*** since the problem p_i^nconsists only on repeating the execution of the same problem p_i ***n*** finite times. Hence, our computer shouldn't consider that $p_i^n$is an infinite loop when ***n*** is bigger than a defined value ***O*** that depends on the computer system otherwise p_i^n becomes an undecidable problem.

We consider also that ***G*** is a fixed number of inputs for all the decidable problems concerned by this work. We know that the difference of complexity time between P-problems and NP-Hard problems increases as ***G*** becomes bigger. And thus, we suppose the existence of a fixed number of inputs ***G*** that keeps any ***P***-problem as a product of a finite number of elements $p_i$which means that ***P***-problems will keep their finite times of execution, but we consider that ***G*** is big enough to make all decidable NP-Hard problems a product of an infinite number

of elements $p_i$which means that the decidable NP-Hard problems will have an infinite time of execution. This means that even if the number **G** is very big, it keeps the strict order of complexity time for ordinary computers.

We consider that this supposition is possible in the computing field and we use this fixed number ***G*** in all the following demonstrations.

All decidable problems can be modelled by algorithms, and we know that the execution time of some algorithms can't end even if the algorithm doesn't contain any infinite loop. We have no solution to these problems and these problems can belong to the sets of ***NP*** problems or NP-Hard problems. There are indeed decision problems that are decidable but in the set of NP-Hard problems because of the time hierarchy theorem.

We will focus here on solving the optimization corresponding to the decision problem (by using a polynomial number of calls to the decision problem).

We can accept that the algorithm execution that doesn't end is equivalent to the execution of an infinite series of simple problems p_i. The NP-Complete problems and the decidable NP-Hard problems have never been solved even if they have algorithms. And thus, the execution times of these algorithms can be considered infinite.

Consequently, we consider that the set of NP-Complete problems is composed of elements c_j with $j \in \mathbb{N}$ and j>0 with:

$$c_j = \prod_{i=1}^{+\infty} p_i \quad (3)$$

and we consider that in the set of NP-Hard problems there are some elements: h_k with $k \in \mathbb{N}$ and k>0 with:

$$h_k = \prod_{i=1}^{+\infty} p_i \quad (4)$$

if h_k is decidable.

However, we can check a solution for any ***NP-***problem in a finite execution time. Hence, let's define an application ***Check*** that gives problems of finite times of execution if it is possible when it is applied to decidable problems. Since ***NP-***problems are all verifiable, then the application ***Check*** gives P problems when it is applied to them.

In this article Check is an application:

from: Set of decidable problems ***to:*** Set of P problems ∪ Set of NP-Hard problems

With: $\text{Check}(c_j) = \prod_{i=1}^{n} p_i$ with $n \in \mathbb{N}$ and n finite (5)

And since $p_i$is a solution of $p_i$then we have also:

$$\text{Check}(p_i)=p_i \quad (6)$$

And also:

$$\text{Check}(p_1 \times p_2) = \text{Check}(p_1) \times \text{Check}(p_2)=p_1 \times p_2 \quad (7)$$

with:$\text{Check}(e)$=e. (8)

However, since we don't know if NP = P or not, it is still hard to say if we can verify a NP-Hard problem in polynomial time or not. Hence, we don't check in this work a solution for NP-Hard problems, and we have to accept that :

$\text{Check}(h_k)=h_k$ with h_k has an infinite time of execution. (9)

We conclude that: $p_i \in$ Set of P problems ⇔ $\text{Check}(p_i)=p_i$ (10)

And:

$c_j \in$ Set of NP-Complete problems ⇔ $\text{Check}(c_j) = \prod_{i=1}^{n} p_i$ with $n \in \mathbb{N}$ and n finite

(11)

This useful result from the considerations will allow us to develop a logical mathematical proof about the sets of complexity.

The compilators of each programmation language can propose programs that can count the number of operations necessary for the execution of a studied algorithm that is equivalent to a problem of a given complexity.

3 A comparison between NP-Complete problems and NP-Hard problems

We remark that:

$$\text{Check}(c_j) = \text{Check}(\prod_{i=1}^{+\infty} p_i) = \prod_{m=1}^{n} p_m \quad \text{with} \quad n \in \mathbb{N} \quad \text{and} \quad \text{n finite} \tag{12}$$

Hence:

$$\prod_{i=1}^{n} p_i \times \text{Check}(\prod_{i=n+1}^{+\infty} p_i) = \prod_{m=1}^{n} p_m \tag{13}$$

We have always:

$$\prod_{i=1}^{n} p_i \neq \prod_{m=1}^{n} p_m \tag{14}$$

otherwise we would have:

$$\text{Check}(\prod_{i=n+1}^{+\infty} p_i)=e \tag{15}$$

which is impossible since we have :

$$\prod_{i=n+1}^{+\infty} p_i \neq e \text{ for all elements } p_i. \tag{16}$$

Let's consider that ***A*** is the inverse of $\prod_{i=1}^{n} p_i$. (17)

Hence we have:

$$\text{Check}(\prod_{i=n+1}^{+\infty} p_i) = \prod_{m=1}^{n} p_m \times A \tag{18}$$

We also have:

$$\prod_{i=n+1}^{+\infty} p_i = \prod_{i=1}^{+\infty} p_{n+i} \tag{19}$$

Consequently:

$$\prod_{m=1}^{n} p_m \times A=\text{Check}(\prod_{i=1}^{+\infty} p_{n+i}) \tag{20}$$

We don't know if the decidable element $\prod_{i=1}^{+\infty} p_{n+i}$ is verifiable (checkable) or not. Hence we consider that ***A*** is an element of the set of NP-Hard problems.

And we can change ***n*** in this method with ***n+l*** with ***l*** is an integer with ***l>-n*** because if we have $p_3=p_1 \times p_2$ then p_3 is also an element of the set of ***P*** problems.

However, by using the new discovered element ***A***, we can discover many other elements h_i of the set of NP-Hard problems that have a bigger complexity only by considering that $h_i=A \times d_i$ where $d_i$can be any element of the set of decidable problems.

Finally we proved that each element c_j of the set of the NP-Complete problems produces many new elements h_i that have an algorithm with an infinite time of execution and that we can't check easily because we can't deduce obviously the algorithm of the problem ***A*** from the algorithm of its inverse that is in with the help of the application ***Check***.

We conclude that each $c_j$produces at least ***D*** elements of the set of the NP-Hard problems with D=card(Set of decidable problems).

And thus:

$$\text{card}(\text{Set of NP-Hard problems}) \geq \text{card}(\text{Set of decidable problems}) \times \text{card}(\text{Set of NP-Complete problems}) \tag{21}$$

Let's remember that there exist many undecidable problems in the set of NP-Hard problems such as the Turing halting problem. For these kinds of problems, there is no algorithm that can answer correctly on all inputs.

4 The cardinal of NP-Complete problems

If we make the product of all the elements p_i of the set of ***P*** problems, then we get a bigger problem **B**

that has an algorithm of an infinite time of execution since the number of the elements of the set of ***P*** problems can be considered infinite. However the problem ***B*** is decidable since it can be modeled by an algorithm that represents the product of problems p_i.

Thanks to the time hierarchy theorem, if we want to reduce that complexity of the problem ***B*** then we have to avoid executing some problems that belong to the product of problems p_i when we are executing the algorithm of problem ***B***.

If we want to remove an element p_i from this product (by multiplying ***B*** with the inverse of p_i) , then we have a number of possibilities equal to card(Set of P problems). (22)

We consider that each possibility of this new product is B_i. If we want to remove an other element p_j from a product B_i (by multiplying B_i with the inverse of p_j) , then we have another number of possibilities equal to card(Set of P problems). This is because the inverse of p_j removes only the element p_j but not the element p_j^n where ***n*** can be any positive natural number. Furthermore, p_j^{n+1} always exists in the the product of problems $p_i$that makes ***B*** since n+1 is also finite.

We consider that each possibility of this new product is B_{ij}.

We repeat the same operation ***L*** times with card(Set of P problems) − L=+∞ (23)

And this operation allows us to create a set ***DC*** of ***N*** decidable elements C_k with 0<k<N+1 and each $C_k = \prod_{i=1}^{+\infty} p_i$ is a problem that has an algorithm with an infinite time of execution, and we have: $N=\left(\text{card(Set of P problems)}\right)^L$ (24)

L can even have an infinite value but in order to have: $C_k \in$ Set of NP-Hard problems ∩ Set of decidable problems (25)

We should always have:

card(Set of P problems) − L=+∞ (26)

We know that when ***L*** increases, the complexity of the problems $C_k$decreases, and thus we can find the smallest value ***M*** that reduces the complexity of all the problems C_k when ***L=M*** such as we get: $C_k \in$ Set of NP-Complete problems . (27)

This means that with ***L=M*** we have Set of NP-Complete problems ⊆ DC. (28)

<u>Consideration:</u>

Despite its high complexity, each NP-Complete problem can be reduced to another NP-Complete problem and vice versa since NP-Complete problems are also NP problems. Hence, let's consider that all NP-Complete problems have the same complexity time that is directly linked to the value of ***M.*** We consider that if the value of ***L*** decreases then all the elements of the set ***DC*** become NP-Hard problems. This is because we consider that all NP-Hard problems that are not NP-Complete problems have higher complexity than NP-Complete problems.

This means that with ***L=M*** we have Set of NP-Complete problems=DC . (29)

In this case, we have:

$$\text{card}\left(\text{Set of NP-Complete problems}\right) = \left(\text{card(Set of P problems)}\right)^M \quad (30)$$

This result is an equation that allows us also to understand the difference between the cardinals of the set of NP-Complete problems and the set of P-problems even if these two sets have both infinite cardinals.

5 The bijection that links P problems and the set of decidable problems

Let's consider that ***PW*** is the power set of the set of ***P*** problems with the decidable problem ***e*** removed.

Let's consider that PW_i with i>0 are the sets that are elements of ***PW*** except the element: {∅} ***(the empty set)*** that we won't use.

Let's consider an application ***G*** defined:

from the set PW \ {∅} to Set of decidable problems

with: $G(PW_i)$ equals the product of all the elements of PW_i.

Since the elements PW_i are all the possible subsets of the set of ***P*** problems, then the elements $G(PW_i)$ are all the possible elements of the set of decidable problems with the decidable problem ***e*** removed.

Hence, the application ***G*** is a bijection between: PW \ {∅}
and the set of decidable problems considered without the element ***e***.

And thus:

$$\text{card}(\text{Set of decidable problems}) - 1 = 2^{\text{card}(\text{Set of P problems})-1} - 1 \quad (31)$$

Consequently:

$$\text{card}(\text{Set of decidable problems}) = 2^{\text{card}(\text{Set of P problems})-1} \quad (32)$$

We can deduce that:

$$\text{card}(\text{Set of P problems}) \leq 2^{\text{card}(\text{Set of P problems})-1} \quad (33)$$

And:

$$\text{card}(\text{Set of NP problems}) \leq 2^{\text{card}(\text{Set of P problems})-1} \quad (34)$$

And also:

$$\text{card}(\text{Set of NP-Complete problems}) \leq 2^{\text{card}(\text{Set of P problems})-1} \quad (35)$$

This result is a new attempt to compare the cardinals of the set of NP-Complete problems and the set of P-problems even if these two sets have both infinite cardinals.

6 First conclusions and remarks

We can deduce from formula (21) and formula (31) that:

$$\text{card}(\text{Set of NP-Hard problems}) \geq 2^{\text{card}(\text{Set of P problems})-1} \times \text{card}(\text{Set of NP-Complete problems}) \quad (36)$$

And thus, we can deduce from formula (35) and formula (36) that:

$$\text{card}(\text{Set of NP-Complete problems}) \leq \frac{\text{card}(\text{Set of NP-Hard problems})}{\text{card}(\text{Set of NP-Complete problems})} \quad (37)$$

Which is equivalent to:

$$\text{card}(\text{Set of NP-Complete problems}) \leq \sqrt{\text{card}(\text{Set of NP-Hard problems})} \quad (38)$$

And we can also conclude from formula (30) and formula (35) that:

$$\left(\text{card}(\text{Set of P problems})\right)^{M} \leq 2^{\text{card}(\text{Set of P problems})-1} \quad (39)$$

where ***M*** is the smallest value that reduces the complexity of the elements of the Set ***DC*** into the complexity of NP-Complete problems. However, we should always have card(Set of P problems) − M=+∞.

Furthermore, the value of $\left(\text{card}(\text{Set of P problems})\right)^{M}$ increases when ***M*** increases.

After this step, we have new formulae that allow us to develop logical mathematical demonstrations about the sets of complexity especially by using their cardinals.

7 Investigating the value of M that reduces the complexity of the elements of the Set DC into the complexity of NP-Complete problems

Now we should compare : $\left(\text{card(Set of P problems)}\right)^{M}$

and: $2^{\text{card(Set of P problems)}-1} = \frac{2^{\text{card(Set of P problems)}}}{2}$

Let's consider that:

$M=\frac{\text{card(Set of P problems)}}{x}$ with x>1. (40)

Hence, we have:

$\left(\text{card(Set of P problems)}\right)^{M} = \left(\text{card(Set of P problems)}\right)^{\frac{\text{card(Set of P problems)}}{x}}$ (41)

And thus:

$\left(\text{card(Set of P problems)}\right)^{M} = \left(\left(\text{card(Set of P problems)}\right)^{\frac{1}{x}}\right)^{\text{card(Set of P problems)}}$ (42)

However $\left(\left(\text{card(Set of P problems)}\right)^{\frac{1}{x}}\right)$ is infinite for any finite positive number x.

We conclude that if $M=\frac{\text{card(Set of P problems)}}{x}$ with x>1,

then we have: $\left(\text{card(Set of P problems)}\right)^{M}$ is much bigger than $2^{\text{card(Set of P problems)}-1}$.

Consequently, we should have: $M<\frac{\text{card(Set of P problems)}}{x}$ $\forall x>1$ and x finite (43)

This easy mathematical result characterizes the point M. However, other characteristics can be deduced about the point M in order to develop easy other demonstrations about the sets of complexity.

8 Second conclusions and remarks

If we make the set ***DC*** by using M with:

$\text{card(Set of P problems)} - M=+\infty$

but $M \geq \frac{\text{card(Set of P problems)}}{x}$

with x is a real positive finite number, then we have:

$\left(\text{card(Set of P problems)}\right)^{M} > 2^{\text{card(Set of P problems)}-1}$

which makes a contradiction.

And thus, we have always: $M<\frac{\text{card(Set of P problems)}}{x}$ $\forall x>1$ and x finite .

We remark that when a decidable problem d_H is expressed as: $d_H = \prod_{i=1}^{H} p_i$ (44)

where ***H*** is a natural number that respects $H=\text{card(Set of P problems)} - N'=+\infty$ (45)

with: $N' \geq \frac{\text{card(Set of P problems)}}{x}>M$ with x is a finite real number with x>1 (46)

Then we have:

$H \leq \frac{\text{card(Set of P problems)}\times(x-1)}{x}$ (47)

we can also have ***x*** bigger that 1 but very close to 1 and consider that $x=1^{+}$, and this can be useful as demonstrated in the final results of this article since it still allows that:

$$\frac{\text{card(Set of P problems)}\times(x-1)}{x} > 0 \text{ .}$$

Furthermore, the problem: $d_H$stays a decidable problem because the number ***H*** exists and we can also have H=+∞. However we should also remark from the previous paragraphs that d_H in this case has a complexity of ***NP-Problems*** since N>M**.**

Theorem:

When a decidable problem d_H is expressed as:

$d_H = \prod_{i=1}^{H} p_i$ ***where*** p_i ***are P problems that have a finite time of execution***

If we have $H \leq \text{card(Set of P problems)} \times \left(1-\frac{1}{x}\right)$ ***where x is a finite real number with x>1***

then the problem $d_H$is a NP-Problem.

Remark: This new theorem about NP-Complete problems doesn't require that $x = 1^{+}$. This theorem can be very useful for the readers who aim to make personal demonstrations about the sets of complexity or to propose a personal solution for the Millenium Problem of Clay Mathematics about complexity problems (P=NP ?).

An artificial intelligence can also be useful to compare and verify any proposed demonstrations to this problem based on principles of informatics theory.

9 Investigation about NP-Problems

NB: This part of the article is a personal attempt to easily finish the demonstration concerning the millennium problem P=NP. This part of the demonstration needs that the existence of the two considered numbers "z" and " n' " be logically possible in the field of computer science by taking in consideration the mathematical logics respected in this article.

Let's consider a decidable problem d_H expressed as

$$d_H = \prod_{i=1}^{H} p_i \quad (48)$$

wherep_i are P problems that have a finite time of execution with:

$$H \leq \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right) \quad \text{where } \mathbf{z>1}. \quad (49)$$

The problem $d_H$is a NP-Problem. Consequently, let's find the cardinal of the set ***DH*** of all the possibilities of the problems d_{HJ}similar to d_H.

Since $H \leq \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)$ for each $d_{\text{HJ}} = \prod_{i=1}^{H} p_i$in ***DH***, then we consider that ***H*** is the integer part of $\text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)$.

Consequently, we write:

$$\text{H}=\left[\text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right] \quad (50)$$

Hence, the number of possibilities of d_{HJ}can be expressed as:

$$\text{D}=\left(\text{card(Set of P problems)}\right)^{\left[\text{card(Set of P problems)}\times\left(1-\frac{1}{z}\right)\right]-J} \quad (51)$$

And thus:

$$\text{card(DH)} = \sum_{\text{J}=0}^{\left[\text{card(Set of P problems)}\times\left(1-\frac{1}{z}\right)\right]-1} \text{card(Set of P problems)}^{\left[\text{card(Set of P problems)}\times\left(1-\frac{1}{z}\right)\right]-J} \quad (52)$$

We remark that: $\text{card(DH)} \geq \text{card(Set of P problems)}$ (53)

However, we defined the Set *DH as a set that respects:* $\text{DH} \subset \text{Set of NP-problems}$ (54)

Which means that: $\text{card(DH)} < \text{card}\left(\text{Set of NP-problems}\right)$ (55)

Which is equivalent to:

$$\text{card}\left(\text{Set of NP-problems}\right) > \sum_{\text{J}=0}^{\left\lceil \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right\rceil -1} \text{card(Set of P problems)}^{\left\lceil \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right\rceil -J} \quad (56)$$

And we conclude that: $\text{card(Set of P problems)} < \text{card}\left(\text{Set of NP-problems}\right)$ (57)

Which means that:

$$P \neq \text{NP} \quad (58)$$

However, since we have: $\text{card(Set of NP problems)} \leq 2^{\text{card(Set of P problems)}-1}$

Then we have in this case:

$$\sum_{\text{J}=0}^{\left\lceil \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right\rceil -1} \text{card(Set of P problems)}^{\left\lceil \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right\rceil -J} < 2^{\text{card(Set of P problems)}-1} \quad (59)$$

And thus we should check if this is a contradiction for any real number z with z>1.

Let's consider that $z=\frac{\text{card(Set of P problems)}}{\text{card(Set of P problems)}-\text{n'}}$ where $\boldsymbol{n'}$ has a natural positive value. (60)

In this case, even if card(Set of P problems) can be considered infinite, we should find a big natural number $\boldsymbol{n'}$ in order to have at least $z=1^{+} > 1$. (61)

and we have : $\left\lceil \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right\rceil - \text{J} = \text{n'} - J$ (62)

And we have: $\left\lceil \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right\rceil - 1 = \text{n'} - 1$ (63)

Hence, if $z=\frac{\text{card(Set of P problems)}}{\text{card(Set of P problems)}-\text{n'}}$ then:

$$\sum_{\text{J}=0}^{\left\lceil \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right\rceil -1} \text{card(Set of P problems)}^{\left\lceil \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right\rceil -J} = \sum_{\text{i}=0}^{\text{n'}-1} \text{card (Set of P problems)}^{\text{n'}-i} \quad (64)$$

And if $\boldsymbol{n'}$ allows also to respect the scale of limits then we have:

$$2^{\text{card(Set of P problems)}-1} > \text{card(Set of NP problems)}^{\text{n'}+1} > \text{n'} \times \text{card(Set of NP problems)}^{\text{n'}} > \sum_{\text{i}=0}^{\text{n'}-1} \text{card (Set of P problems)}^{\text{n'}-i} \quad (65)$$

And thus, if $z=\frac{\text{card(Set of P problems)}}{\text{card(Set of P problems)}-\text{n'}}$ where $\boldsymbol{n'}$ exists,

then this formula stays correct:

$$\sum_{\text{J}=0}^{\left\lceil \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right\rceil -1} \text{card(Set of P problems)}^{\left\lceil \text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right\rceil -J} < 2^{\text{card(Set of P problems)}-1} \quad (66)$$

However we should choose the appropriate number $\boldsymbol{n'}$ that is big enough to be significant compared to card(Set of P problems) in order to verify the formula (61) and the formula (66) at the same time.

We conclude that if we find the appropriate real number $\boldsymbol{z}$ that prevents formula (66) from being a contradiction, then we can accept this conclusion:

Final conclusion depending on the taken considerations:

We have: $\text{card(Set of P problems)} < \text{card}\left(\text{Set of NP-problems}\right)$

And thus: $P \neq$ NP**.**

10 Conclusion

The results of this work are made by considering that:

1) A fixed number of inputs G exists and can be defined in the field of computing such as G keeps any P-problem as a product of a finite number of P-problems p_i, but G is big enough to make all decidable NP-Hard problems a product of an infinite number of elements p_i.

2) All NP-Hard problems that are not NP-Complete problems have higher complexity than NP-Complete problems and by considering that all NP-Complete problems have one same complexity that is directly linked to the value of a number M that should exist and which is a fixed number of P-Problems. M should have a natural value that respects: $M<\frac{\text{card(Set of P problems)}}{x}$ $\forall x>1$ and x finite**.**

3) The appropriate real number z that prevents formula (66) from being a contradiction exists and can be defined in the computing field.

We proved in this article by using these considerations that:
$\text{card}(\text{Set of NP-Hard problems}) \geq$
$\text{card}(\text{Set of decidable problems}) \times$
$\text{card}(\text{Set of NP-Complete problems})$

And also: $\text{card}(\text{Set of NP-Complete problems}) = (\text{card}(\text{Set of P problems}))^{M}$

where ***M*** is the smallest value that reduces the complexity of the elements of the Set ***DC*** presented above into the complexity of NP-Complete problems.

However, we should always have $\text{card}(\text{Set of P problems}) - M = +\infty$.

We proved also that:
$\text{card}(\text{Set of decidable problems}) = 2^{\text{card(Set of P problems)}-1}$

Hence, we deduced that: $\text{card}(\text{Set of P problems}) \leq 2^{\text{card(Set of P problems)}-1}$

And: $\text{card}(\text{Set of NP problems}) \leq 2^{\text{card(Set of P problems)}-1}$

And also: $\text{card}(\text{Set of NP-Complete problems}) \leq 2^{\text{card(Set of P problems)}-1}$

We concluded that:

$\text{card}(\text{Set of NP-Hard problems}) \geq 2^{\text{card(Set of P problems)}-1} \times \text{card}(\text{Set of NP-Complete problems})$

And that: $\text{card}(\text{Set of NP-Complete problems}) \leq \frac{\text{card}(\text{Set of NP-Hard problems})}{\text{card}(\text{Set of NP-Complete problems})}$

Which is equivalent to:

$$\text{card}(\text{Set of NP-Complete problems}) \leq \sqrt{\text{card}(\text{Set of NP-Hard problems})}$$

We could also conclude that:

$(\text{card}(\text{Set of P problems}))^{M} \leq 2^{\text{card(Set of P problems)}-1}$

with always $M<\frac{\text{card(Set of P problems)}}{x}$ $\forall x>0$ and x finite

And we proved that:

$$\text{card(Set of NP-problems)} > \sum_{J=0}^{\left\lceil \text{card(Set of P problems)}\times\left(1-\frac{1}{z}\right)\right\rceil-1} \text{card(Set of P problems)}^{\left\lceil \text{card(Set of P problems)}\times\left(1-\frac{1}{z}\right)\right\rceil-J}$$

<u>Finally we concluded this theorem:</u>

When a decidable problem d_H is expressed as:

$d_H = \prod_{i=1}^{H} p_i$ ***where*** p_i ***are P problems that have a finite time of execution***

If x is a finite real number with x>1, then we have:

$$\left(\text{x is a finite real number with x>1} \quad \text{and} \quad H \leq \text{card(Set of P problems)} \times \left(1 - \frac{1}{x}\right)\right) \Rightarrow d_H \text{ is a NP-Problem}$$

<u>And if we can define in the field of computer science the considered number "z" that prevents the following formula from being a contradiction:</u>

$$\sum_{J=0}^{\left[\text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right]-1} \text{card(Set of P problems)}^{\left[\text{card(Set of P problems)} \times \left(1-\frac{1}{z}\right)\right]-J} < 2^{\text{card(Set of P problems)}-1}$$

<u>then this allows to make this conclusion:</u>

We have: $\text{card(Set of P problems)} < \text{card}\left(\text{Set of NP-problems}\right)$

And thus: $P \neq \text{NP}$***.***

NB: the value of the number "M" can be studied further in order to find new mathematical characteristics that allow finding a solution to the problems of complexity sets without needing to find the considered numbers "z" and " n' ".

<u>Final conclusion:</u>

The Millenium problem (P=NP ?) is not only a problem of informatics but also a logical philosophical problem. A solution to this Clay Mathematics problem will allow us to classify logical problems better in order to find the suitable computer for each kind of problems which will allow us to enhance the efficiency of all the fields that need informatics.

This proof can also be useful as a basis for the researchers who deal with Quantum Computers. Furthermore, the value of the demonstrated number "M" can be studied further in order to find a solution to this Millenium problem without needing to find the considered number "z".

The readers are invited to this opportunity in order to investigate the proposed logical considerations for a solution to this important Clay Mathematics Millenium problem.

References:

[1] akram louiz. Complex mathematical statements useful as criterion for Landau-Siegel Zeros, 29 January 2023, PREPRINT (Version 2) available at Research Square. DOI: https://doi.org/10.21203/rs.3.rs-2520944/v2

[2] Hidalgo-Herrero, M., Rabanal, P., Rodriguez, I., & Rubio, F. (2013). Comparing problem solving strategies for NP-hard optimization problems. *Fundamenta Informaticae*, *124*(1-2), 1-25, 2013, DOI: https://doi.org/10.3233/FI-2013-822

[3] Izadkhah, H. (2022). P, NP, NP-Complete, and NP-Hard Problems. In *Problems on Algorithms: A Comprehensive Exercise Book for Students in Software Engineering* (pp. 497-511). Cham: Springer International Publishing, 2022. DOI: https://doi.org/10.1007/978-3-031-17043-0_15

[4] Alizadeh, R., Allen, J.K. & Mistree, F. Managing computational complexity using surrogate models: a critical review. *Res Eng Design* 31, 275–298, 2020. DOI:

https://doi.org/10.1007/s00163-020-00336-7

[5] Savitch, W. J. (1970). Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, *4*(2), 177-192, 1970. DOI: https://doi.org/10.1016/S0022-0000(70)80006-X

[6] Paul, W. J., Pippenger, N., Szemeredi, E., & Trotter, W. T. (1983, November). On determinism versus non-determinism and related problems. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)* (pp. 429-438). IEEE, 1983. DOI: https://doi.org/10.1109/SFCS.1983.39

[7] ZAK, S. (1983). A turing machine time hierarchy. *Theoretical computer science*, *26*(3), 327-333, 1983. DOI: https://doi.org/10.1016/0304-3975(83)90015-4

[8] Freedman, M. H. (1998). P/NP, and the quantum field computer. *Proceedings of the National Academy of Sciences*, *95*(1), 98-101, 1998. DOI: https://doi.org/10.1073/pnas.95.1.98

[9] Gharibian, S., Huang, Y., Landau, Z., & Shin, S. W. (2015). Quantum hamiltonian complexity. *Foundations and Trends® in Theoretical Computer Science*, *10*(3), 159-282, 2015. DOI: https://doi.org/10.1561/0400000066

[10] Ohya, M., & Masuda, N. (2000). NP problem in quantum algorithm. *Open Systems & Information Dynamics*, *7*(1), 33-39, 2000. DOI: https://doi.org/10.1023/A:1009651417615

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The author contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The author has no conflict of interest to declare that is relevant to the content of this article.