https://doi.org/10.31871/WJRR.13.3.8

World Journal of Research and Review (WJRR)
ISSN: 2455-3956, Volume-13, Issue-3, September 2021 Pages 21-26

Review of the Solutions of the Clay Millennium Problem about P $\neq$ NP =EXPTIME

Konstantinos E. Kyritsis

***Abstract*— In this paper I review the decisive proofs that P $\neq$ NP, and NP=EXPTIME in the context of the Zermelo-Frankel set theory and deterministic Turing machines. The results of these proofs definitely solve the 3rd Clay Millennium Problem about P versus NP, in a simple and transparent away that the general scientific community, but also the experts of the area, can follow, understand and therefore become able to accept. So far the solutions of this famous problem seem that they have not been noticed by the wider relevant scientific community. The main purpose of this paper is to make widely known the solutions of the 3rd Clay Millennium problem "P versus NP" to the relevant scientific community.**

***Index Terms*—3rd Clay Millennium problem, EXPTIME-complete problems, NP-complexity, P-complexity**

Mathematical Subject Classification: 68Q15.

I. Introduction

Two solution of the famous P versus NP problem has been published in [9] Kyritsis C and in this paper we review, discuss and present the solutions. In the history of mathematics, it is known that difficult problems that have troubled a lot the mathematicians, turned out to have different proofs one simple and one very complex. Such an example is if the general 5th order polynomial equation can be solved with addition, subtraction, multiplication, division and extraction of radicals starting from the coefficients. The famous mathematician Niels Henrik Abel who gave a very simple proof. On the other hand, the proof of the same, by the E. Galois theory, is a whole book of dozens of pages!

And a famous mathematician once said that "Once a proof is known to a mathematical problem, then immediately after it becomes trivial!"

It is the same with the solution of the P versus NP problem in this paper. We will utilize in our proofs, the key abstraction of the existence of an EXPTIME complete language, (it is known that it exists) without specifying which one, which will simplify much the arguments. Then we synthesize other languages and arguments over it, that will solve the problem.

A second issue that is important to mention, is a statement, that is usually attributed to the famous mathematician Yuri

Manin, that "A correct proof in mathematics is considered a proof only if it has passed the social barrier of being accepted and understood by the scientific community and published in accepted Journals"

Passing the obstruction of the social barrier, sometimes is more difficult than solving the mathematical problem itself!

We must notice here that the P versus NP problem, is in fact a set of different problems within different axiomatic systems. In the context of what axiomatic system is the Complexity Theory of Turing machines? Since the complexity theory of Turing machines requires entities like infinite sets of words then it is in the context of some axiomatic set theory, together with the axiom of infinite. So we notice that the next are different problems:

1) The P versus NP problem in the Zermelo-Frankel axiomatic system of sets without the axiom of choice and this axiomatic system formulated in the 2rd order formal languages.
2) The P versus NP problem in the Zermelo-Frankel axiomatic system of sets with the axiom of choice and this axiomatic system formulated in the 2rd order formal languages.
3) Etc

We might try to think of the P versus NP problem within the context of the axiomatic system of Peano Arithmetic with or without the axiom of induction and within second order formal languages. But to do so, we must carefully define, what additional axioms or definitions give the existence of infinite subsets of natural numbers that are used in the Complexity Theory.

The main hidden guiding idea in searching for such a simple proof, was that what the "arbitrary human-like free-will" of a non-deterministic Turing machine as human-machine interactive software (e.g. in password setting), can do in polynomial time cannot be done by a purely mechanical deterministic Turing machine in polynomial time. (See also beginning of paragraph 4) After the Key-abstraction mentioned above I had to find the right simple argumentsto make a valid proof of this idea. The proof of the P versus NP problem in the direction P $\neq$ NP, is supposed also to mean that the standard practice of encryption in the internet, is safe.

We notice also that the P versus NP:

1) It is a difficult problem, that has troubled the scientific community for some decades
2) It may have simple proofs of a few paragraphs, hopefully not longer than the proof of the Time Hierarchy theorem, which seems to be a deeper result.
3) But it can also have very lengthily and complex proofs, that may take dozens of pages.

4) There many researchers (more than 5 of them) that have claimed to have solved it, either as P=NP, or as P $\neq$ NP, and

Konstantinos E. Kyritsis Dept. Accounting-Finance University of Ioannina, Greece

even as suggestion that neither are provable, but only a handful of them seem to have been able to pass the preliminary social barrier and publish their solution in conferences or Journals with referees. The rest of them have published online only preprints (see e.g. the [17] P versus NP page). It seems to me though that it is not probable that all of them have correct solutions. Especially in the direction P=NP, there is a common confusion and mistake, that has been pointed out by Yannakakis M. [18]. Furthermore, this confusing situation has contributed so that although there are publications in respectable Journals, the experts and the scientific community does not seem of being able to decide if the P versus NP problem has been solved or not. This is reasonable, as there are proofs of close to 100 pages, and no average reader would feel comfortable to go through them, and decide for himself if there a flaw or error somewhere. Still it is better to have published results than non-published, and then let the large number of readers to try to find errors or flaws in the solutions if there are any.

So here comes the need of a more challenging problem: Not only to solve the P versus NP problem, but also solve it in such an simple, elegant and short way, so that the researchers will know a decisive proof that they can understand and control that P ≠ NP or not, so short that anyone familiar with the area, would discover any flaw or error if it existed.

This is I believe the value of the present paper that provides such a proof in the context of the Zermelo-Frankel set theory (we do not use the axiom of choice), formulated e.g. within 2nd order formal languages.

What this proof is or is not:

1) It does not introduce new theoretical concepts in computational complexity theory so as to solve the P versus NP.
2) It does not use relativization and oracles
3) It does not use diagonalization arguments, although the main proof, utilizes results from the time hierarchy theorem
4) It is not based on improvements of previous bounds of complexity on circuits
5) It is proved with the method of counter-example. Thus it is transparent short and "simple". It takes any Exptime-complete DTM decision problem, and from it, it derives in the context of deterministic Turing machines a decision problem language which it is apparent that it belongs in the NP class decision problems while it does not belong the class P of decision problems.
6) It seems a "simple" proof because it chooses the right context to make the arguments and constructions and the key-abstraction mentioned above. So it helps the scientific community to accept that this 3rd Clay Millennium problem has already been solved.

In the paragraph 4, we give an advanced, full proof that P ≠ NP, in the standard context of deterministic Turing machines, solving thus the 3rd Clay Millennium problem.

II. PRELIMINARY CONCEPTS, AND THE FORMULATION OF THE 3RD CLAY MILLENNIUM PROBEM, P VERSUS NP.

In this paragraph, for the sake of the reader, we will just mention the basics to understand the formulation of the 3rd Clay Millennium problem. The official formulation is found in [3] (Cook, Stephen *(April 2000), The P versus NP Problem* (PDF), *Clay Mathematics Institute site)*. Together with an appendix where there is concise definition of what are the Deterministic Turing machines, that is considered that they formulate, in Computational Complexity theory , the notion and ontology of the software computer programs.

In the same paper are also defined the computational complexity ***classes P, NP***.

The elements of the classes P, NP etc strictly speaking are not only sets of words denoted by L, that is not only languages, but also for each such set of words or language L at least one DTM , M that decides it, in the specified complexity so they are pairs (L,M). Two such pairs (L_1, M_1) (L_2, M_2) are called ***equidecidable*** if $L_1 = L_2$ although it may happen that $M_1 \neq M_2$. E.g. if the complexity of M_1 is polynomial-time while that of M_2 exponential-time choosing the first pair instead of the second means that we have turned a high complexity problem to a feasible low complexity problem.

The definition of other computational complexity classes like **EXPTIME** etc. can be found in standard books like [6],[11],[12]. In the official formulation [3] there is also the definition of the concept of ***a decision problem language in polynomial time reducible to another decision problem language***.

Based on this definition it is defined that an EXPTIME-complete decision language of EXPTIME is EXPTIME-complete, when all other decision problems languages of EXPTIME have a polynomial time reduction to it. Here is the exact definition

Definition 2.1 *Suppose that Li is a language over all words Σ_i , $i = 1, 2$. Then $L_1 \leq p \quad L_2$ (L_1 is p-reducible to L_2) iff there is a polynomial-time computable function $f : \Sigma_1 \to \Sigma_2$ such that $x \in L_1$ if and only if $f(x) \in L_2$, for all $x \in \Sigma_1$.*

In the same books [6],[10],[11] can be found the concepts and definitions of ***NP-complete and EXPTIME-compete decision problems***. See also [7], [12] where its proved that specific decision problems are EXPTIME-complete.

For simplicity we will consider here only binary alphabets {0,1} and sets of binary words Σ.

III. WELL KNOWN RESULTS THAT WILL BE USED.

We will not use too many results from the computational complexity theory for our proof that P ≠ NP.

A very deep theorem in the Computational Complexity is the ***Time Hierarchy Theorem*** (see e.g. [6],[11],[12],[10],[14]. This theorem gives the existence of decision problems that cannot be decided by any other deterministic Turing machine in less complexity than a specified.

Based on this theorem, it is proved that:

Proposition 3.1 *There is at least one EXPTIME-complete decision problem, that cannot be decided in polynomial time, thus $P \neq EXPTIME$.*

The next two propositions indicate what is necessary to prove in order to give the solution of the P versus NP problem.

Proposition 3.2 *If the class NP contains a language L which cannot be decided with a polynomial time algorithm, then $P \neq NP$.*

Proposition 3.3 *If the class NP contains a language L which is EXPTIME complete, then NP=EXPTIME.*

IV THE SOLUTION: P ≠ NP=EXPTIME IN THE CONTEXT OF DETERMINSITIC TURING MACHINES

We will prove in this paragraph that $P \neq NP$ in the context of second order formal language of the Zermelo-Frankel set theory.

Since we are obliged to take strictly the official formulation of the problem, rather than text books about it, we make the next clarifications.

We will use the next conditions for a Language to be in the class NP, as stated in the official formulation of the P versus NP problem (see **[3] Cook, Stephen *(April 2000)***, *The P versus NP Problem (PDF), Clay Mathematics Institute.).*

We denote by Σ^* *all the words of an alphabet* Σ.

Definition 4.1 *A language L of binary words is in the class NP if and only if the next conditions hold*

1) *There is a deterministic Turing machine M that decides L. In other words for any word x in L, when x is given as input to M, then M accepts it and if x does not belong to L then M rejects it.*

In symbols: ∃ a deterministic Turing machine M, such that □ $x \epsilon \Sigma^*$, x is either accepted or rejected by M and if M accepts $x \rightarrow x \epsilon L$, and if M reject $x \rightarrow x \notin L$

2) *There is a polynomial-time checkable relation R(x,y), and a natural number k of N, so that for every word x , x belongs to L if and only if there is a word y , with $|y| <= |x|^k$, and R(x,y) holds.*

In symbols:∃ relation R which is polynomial-time checkable ,and ∃ kεN, such that □ $x\epsilon\Sigma^*$,
$x\epsilon L \leftrightarrow (\exists\, y\epsilon\, \Sigma^*$, $|y| <= |x|^k$ and R(x,y) holds).

Remark 4.1. In the official statement of the P versus NP problem (see **[3] Cook, Stephen *(April 2000)***, *The P versus NP Problem (PDF), Clay Mathematics Institute)* the condition 1) is not mentioned. But anyone that has studied complexity theory, knows that it is required. The condition 2) alone cannot guarantee that there is a deterministic Turing machine that decides the language., as the polynomial checkable relation works only if we provide it with certificate y, and not with only x as input. Indeed we shall see below at the end of the proposition in **Remark 4.4,** that there is even an undecidable language L , for which nevertheless there is a polynomial checkable relation R, so that condition R is satisfied. The languages of NP cannot be semidecidable (or undecidable). The NP class is also defined as NP $=\cup_{k\epsilon N}$ NTIME(n^k), but this definition is also in the context of non-deterministic Turing Machines. The situation with P, is more clear, because the mere requirement that a language of P is of polynomial time complexity as it is standard to define it , involves already that there exist a deterministic Turing machines that for every input word, it halts within polynomial time steps and either accepts or rejects it, therefore it decides it. And not only that is simply the language of a deterministic Turing machine , and therefore maybe only semi-decidable.

Remark 4.2. Notice that in the condition 2) the k depends on the relation R and is not changing as the certificate y changes. In other words k does not depend on y and we ***did not*** state the next:

There is a polynomial-time checkable relation R(x,y), so that for every word x , x belongs to L if and only if there is a word y , and k in N ,with $|y| <= |x|^k$, and R(x,y) holds. In symbols: ∃ relation R which is polynomial-time checkable , such that □ $x\epsilon\Sigma^*$, $x\epsilon L \leftrightarrow (\exists y\epsilon\Sigma^*$ and $\exists k\epsilon N$ such that $|y| <= |x|^k$ and R(x,y) holds).

In the official statement of the P versus NP problem (see [3] Cook, Stephen *(April 2000), The P versus NP Problem (PDF), Clay Mathematics Institute)* this is not made clear, in the natural language that the definition is stated. But that k does not depend on the certificate, but on the polynomial checkable relation becomes clear, when we look at the proof in any good textbook about complexity theory, of how a non-deterministic Turing machine which runs in polynomial time, can define a deterministic Turing machine with a polynomial time checkable relation, which is considered that replaces it.

Remark 4.3: My main intuition to find a proof that *$P \neq NP$=EXPTIME. The* **password setting.**

Let us make the next thought-experiment: Imagine a human Mr H who has available infinite time, and has infinite mental capabilities. No the world asks Mr H to set passwords on all lengths of words! So Mr H sets a password p(l) for words of length l=1,2, 3,...n,...etc. Next let us imagine the problem of finding the password of length say x=153. Mr H has an arbitrary free will and he is honest not to give his passwords, in addition Mr H has provided us with a device D(l) for each length l, that unlocks if we give to it the password p(l), so we will know if w is the password or not. So the only way to discover if particular word w of length |w|= l=153 is the password p(153) or not, it is to search all the words of length l in an exhaustive way and try them on the device D(l). This is of course a an EXPTIME complexity problem, that cannot be reduced to a polynomial time problem. Therefore, finding the language LP of passwords p(l) of Mr H, cannot be a problem of polynomial time complexity. If in addition, we assume that the blind *exhaustive search of all words of length l, is an EXPTIME-complete complexity problem on the initial data l, then finding the language of passwords of Mr H is also an EXPTIME-complete problem. Nevertheless, for each word w of length l, the Device(l) is the polynomial time on the length l, checkable relation (certificate for each word w), that can decide if w is a password or not, therefore the problem of finding the language L, is in the NP class complexity. But the above then after Propositions 3.2 and 3,3, indicate that P ≠ NP=EXPTIME.*

Now this intuitive idea, is obviously not a formal proof at all, as we are taking about "human Mr H", "arbitrary human free will" etc. Besides we are taking about the complexity of problems here rather than the complexity of languages. How can we turn this intuition to strict and formal proof, without using oracles, or non-formal arguments? The solution is the key-abstraction that I mentioned in the Introduction, that is to start with the existence of an EXPTIME-complete complexity language, that we know it exists, without specifying which one. Then define other languages over it and make simple arguments that solve to P versus NP problem.

So the strategy to solve the P versus NP problem, is quite simple: We will start with an exptime-complete decision problem and its language L_{exp} and we will derive from it an NP class decision problem than cannot be solved in the polynomial time (it does not belong to the class P).

The next proposition sets the existence of an EXPTIME-complete complexity language of the EXPTIME complexity class (Proposition 3.1) in a convenient form, that can be used for further compositions of other languages over it.

Proposition 4.1. *There is at least one infinite binary sequence, that can be computed and decided as an exptime-complete complexity.*

Proof.

Let an exptime-complete decision problem A , that its existence is guaranteed by **Proposition 3.1** ,and its language L_{exp} ϵ EXPTIME. We will need for the sake of symbolic convenience this language and decision problem , in the form of a binary sequence. If Σ^* is the set of all words of the binary alphabet Σ of the language L_{exp}, then we give a linear order to the binary alphabet Σ={0,1} 0<1, and then the inherited linear lexicographic order to the set of words Σ^*. Since Σ^* is linearly and well ordered with a first element and after excluding all words with a left sequence of consecutive zeros (which is obviously a polynomial time decision on the length of the words) reducing to the set denoted by Σ^{**} ,we fix the identity map as an arithmetization with an 1-1 and on to correspondence F: Σ^{**} -> **N** to the set of natural numbers, so that the language L_{exp} can be considered after this fixed arithmetization identity mapping correspondence F, as a subset of the natural numbers. So let Char(L_{exp}) : **N**->{0,1} be the characteristic function of the set L_{exp} in the Natural numbers encoded thus in a binary base. Then Char(L_{exp}) consists of d_i, for i ϵ **N**, and d_i is binary digit, that is equal to 0 or 1. A first finite 7-digits segment of it, would seem for example like (0010110...). Since L_{exp} is an exptime-complete decision problem L_{exp} ϵ EXPTIME, its characteristic function is computable with an exptime-complexity too on the length of the binary words , and conversely any Turing machine computation of this characteristic function and also infinite binary sequence Char(L_{exp}) : **N**->{0,1}: consisting from d_i/ for all i ϵ **N**, and d_i is binary digit, that is equal to 0 or 1, is also a Turing machine decision computation of the language L_{exp}. Therefore, there is no polynomial time complexity computation of this infinite binary sequence, as this would make EXPTIME=P and we know that P ≠EXPTIME. For the sake of intuitive understanding of the following arguments we call this binary sequence "*An exptime-compete binary DNA sequence*" and we denote it by DNA_{exp}. This simplification from the original exptime-complete decision problem and Language L_{exp} of Σ^* to the DNA_{exp} of N can be considered also as a polynomial time reduction of decision problem and languages $L_{exp} \leq p$ DNA_{exp} *(*L_{exp} *is p-reducible to* DNA_{exp})(see Definition 2.1). QED.

Proposition 4.2 (1st solution) (3rd Clay Millennium problem) *There is at least one decision problem language of the class NP which is not also in the class P. Therefore, P ≠ NP.*

Proof.

In the next we show that there is a language L_{np} belonging to the class NP that cannot also belong to the class P without making, the previous binary sequence in the proof of the Proposition 4.1 called "*exptime-complete binary DNA sequence*" and denoted by DNA_{exp}, computable in polynomial time complexity! To ensure that a language L_{np} belongs to the class NP it must hold that there is a polynomial-time checkable relation R(x,y) and a natural number k, so that for every word x, it holds that x belongs to the language L_{np} if and only if there is another word y, called "certificate" with length $|y|<=|x|^k$,so that R(x,y) holds. Here by |x| we denote the length of the word x, which is a natural number.

Now comes the intuition behind calling the binary sequence DNA_{exp} of the previous proof, a DNA sequence: The trick here is to define this language denoted by L_{np} with the information encoded in the binary sequence DNA_{exp} so that, although a human with deterministic Turing machines and exptime-time complexity can compute DNA_{exp} and therefore decide L_{np} , no deterministic Turing machine within polynomial-time complexity can compute and decide the L_{np}. In addition for every word x, if a human will give to such deterministic machines the necessary information in the form of a "certificate" y, then a deterministic Turing machine can decide if x belongs or not to L_{np} within polynomial-time complexity.

We define such a language L_{np} with the previous requirements simply as follows:

For any word x ϵ Σ^* ,x ϵL_{np} if and only if , the word is an initial word of the infinite binary sequence DNA_{exp} and the $d_{|x|}$ =1 , where $d_{|x|}$ is the |x|-order binary digit of the infinite binary sequence DNA_{exp} . And of course x does not belong to L_{np} if and only if this does not happen. In the original publication of the proof in [9] Kyritsis C. it is not stated explicitly the condition that " the word is an initial word of the infinite binary sequence DNA_{exp}" but it is assumed implicitly as it is apparent from the flow of the arguments, and also as it is stated explicitly in the 2nd alternative proof.

If we re-phrase the condition "$d_{|x|} =1$", as "DNA_{exp} acceptable" , then the definition can be rephrased as that a word x belongs to the language L_{np} if its length and itself is DNA_{exp} acceptable.

Then we define as "certificate" y of the word x, the finite sequence $y=(d_1, d_2,...,d_{|x|})$, and as polynomial time checkable relation R(x,y), and that R(x,y) holds , the fact that given x , and y, the last digit of y is 1 and the rest of the digits agree too. Notice that here a human gives a lot of information to a Turing machine that will check if x belongs or not to L_{np} , in the form of the |x|-length initial segment y of the infinite binary sequence DNA_{exp} that we know that no Turing machine can compute within polynomial-time complexity.

That this relation R(x,y) is checkable in polynomial time relative to the length |x| of x, is obvious as the Turing machine with input x and y, will have only go through |x|-many steps to check the last digit of y.

Now no deterministic Turing machine M can decide the language L_{np} , in other words decide given as input only the word x (without its "certificate" y), if x ϵL_{np} or not. And this is so, because if it exist such a deterministic Turing machine M, then it could also decide (or compute) the digit $d_{|x|}$ of DNA_{exp} which we know that is not computable in polynomial-time complexity. Thus L_{np} does not belong to P, and therefore $P \neq NP$ QED.

Proposition 4.3 (2nd solution) (3rd Clay Millennium problem) *There is at least one decision problem language of the class NP which is not also in the class P. Therefore, $P \neq NP$.It holds also that Np=EXTIME.*

Proof

We may define, in a simpler way, the language $L_{passwords}$ (the index, passwords , is so as to follow the intuition of password setting as in the **Remark 4.3**) as the set of all binary words, that are the successive n-initial segments of the infinite binary sequence $DNA_{exp.}$ Then this language is obviously (after **Proposition 4.1**) an EXPTIME-complete language. Nevertheless the language $L_{passwords}$ also belongs to the class NP, because for each word w, of length |w|=n, a "certificate" y of it is the word w itself y=w, and the polynomial time checkable relation R(w,y) , y=w , is checkable in polynomial time, relative to the length |w|=n. Notice that we have here one only word w for each word-length n. But then from the **Proposition 3.3** NP=EXPTIME, and thus $P \neq NP$. That NP=EXPTIME is direct from the **Proposition 3.3,** and that the language $L_{passwords}$ in the current **proof** of the current proposition, is also EXPTIME-complete language , besides belonging in the class NP. QED.

The next table 1 compares the two solutions

Table 1

Comparisons	**1st solution**	**2nd solution**
Length	**longer**	**shorter**
Use of proposition 4.1	**Yes**	**Yes**
Proves also NP=EXPTIME	**No**	**Yes**

Remark 4.4 Notice that instead of taking, the characteristic function DNA_{exp} of an exptime-complete language, we could have taken the characteristic function DNA_{und} of an undecidable language and we know that, there is at least one, and repeat the definition of the Language L_{np} , deriving thus an undecidable language , not belonging of course to the class NP , which still it has a polynomial time checkable relation, that nevertheless works only if a human feeds it with a certificate y and there is not a Turing machine that can decide it by taking as input the word x alone. This confirms that in the definition of NP in, Definition 4.1, the condition 1) is required. Alternatively we may prove the same thing in a different way. By using the axiom of choice of the ZFC set theory we may define for example an arbitrary infinite sequence L_p of passwords p_n , each one of length exactly n, from the infinite set of the sets of words Σ^n of length n. It is known that the axiom of choice of the ZFC set theory, gives no information at all about what are the elements of such a set, besides that each p_n belongs to Σ^n . We cannot expect that any such infinite choice L_p of n-length passwords p_n can be decided by a deterministic Turing machine. If it was so, as such Turing machines are countable, we order all such languages $L_{p,i}$,$i\epsilon N$ in a sequence and with the diagonal method we define a new and different such language L_0 of passwords , differing to at least one password from all those $L_{p,i}$, thus this L_0 is undecidable (thus not belonging of course to the class NP) . Still again there is a polynomial-time checkable relation $R(x,p_{|x|})$, which simply is checking if $x= p_{|x|}$,so that for every word x, there is a "certificate", here the password $p_{|x|}$, and x belongs to the language L_0 of passwords iff $R(x,p_{|x|})$ holds.

IV. CONCLUSIONS

Sometimes great problems have relatively short and elegant solutions provided we find the **key-abstractions** and convenient context , symbols and semantics to solve them. It requires also a certain power of thinking rather than complexity of thinking, in areas where traditionally and collectively it may not exist before. Even relatively simple paths of reasoning, may be difficult to travel, if there is not, at a certain point of them, the necessary "bridge", that is the necessary key-abstraction or the right conceptual "coins" of symbols and semantics to exchange and convert. Here the key-abstraction was to start from the class EXPTIME and an EXPTIME-complete language of it, without specifying which one instead starting from the class NP. If the P versus NP problem is researched without a main strategy, that would require a short proof, it might become a very complex problem to solve. ***The main hidden guiding idea in searching for such a simple proof, was that, what the "arbitrary human-like free-will" of a non-deterministic Turing machine as human-machine interactive software (e.g. in password setting), can do in polynomial time, cannot be done by a purely mechanical deterministic***

Turing machine in polynomial time. In other words the human-like non-deterministic arbitrariness in Turing machines has an exponential nature. Since in my opinion the Hierarchy Theorem is a deeper result than the P versus NP problem, in principle there should exist a not much more complicated proof of the P versus NP problem, compared to the proof of the Hierarchy Theorem. ***The proof of the P versus NP problem in the direction P ≠ NP, is supposed also to mean that the standard practice of password setting in the internet, is safe when the encryptions is not corrupted and the publicly available hardware computational power is the same for all .***

REFERENCES

[1] Conway J.H. *On numbers and games*, Academic press 1976

[2] Cook, Stephen A. (1972**)**. *"A hierarchy for nondeterministic time complexity"*. Proceedings of the fourth annual ACM symposium on Theory of computing. STOC '72. Denver, Colorado, United States: ACM. pp. 187–192

[3] Cook, Stephen *(April 2000)*, *The P versus NP Problem (PDF)*, *Clay Mathematics Institute site*.

[4] Diduch Rodrigo Gilberto (2012), *P vs NP*, International Journal of Computer Science and Network Security (IJCSNS) Volume 2, pp 165-167.

[5] Gram Seenil 2001 *"Redundancy, Obscurity, Self- Containment & Independence"* by The 3rd International Conference on Information and Communications Security (ICICS 2001) took place in Xian, China, November 13-16, 2001. Proceedings of ICICS as Volume 2229 of Springer Lecture Notes in Computer Science. Pages 495-501.

[6] Harry R. Lewis and Christos H. Papadimitriou *Elements of the Theory of Computation*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981, ISBN 0-13-273417-6.

[7] Hartmanis, J.; Stearns, R. E. *(1 May 1965). "On the computational complexity of algorithms"*. Transactions of the American Mathematical Society. *American Mathematical Society.* ***117****: 285–306.* . ISSN 0002-9947. JSTOR 1994208. MR 0170805.

[8] Kyritsis C**.** *On the solution of the 3rd Clay Millennium problem. A short and elegant proof that P ≠ NP in the context of deterministic Turing machines and Zermelo-Frankel set theory.* Proceedings of the first ICQSBEI 2017 conference, Athens, Greece, pp 170-181

[9] Kyritsis C THE SOLUTION OF THE 3RD CLAY MILLENNIUM PROBLEM. A SHORT PROOF THAT P ≠ NP=EXPTIME IN THE CONTEXT OF ZERMELOFRANKEL SET THEORY. *International Journal of Pure and Applied Mathematics Volume 120 No. 3 2018, pp497-510 ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version) url:http://www.ijpam.eu doi: 10.12732/ijpam.v120i3.1*

[10] Luca Trevisan, *Notes on Hierarchy Theorems*, U.C. Berkeley.

[11] John C. Martin *(1997). Introduction to Languages and the Theory of Computation (2nd ed.).* McGraw-Hill. ISBN 0-07-040845-9.

[12] Papadimitriou Christos (1994). *Computational Complexity.* Addison-Wesley. ISBN 0-201-53082-1.

[13] Rustem Chingizovich Valeyev 2013 *The Lower Border of Complexity of Algorithm of the Elementary NP-Complete Task (The Most Condensed Version)* World Applied Sciences Journal 24 (8): 1072-1083, 2013 ISSN 1818-4952 © IDOSI Publications, 2013.

[14] Stanislav, Žák (October 1983). *"A Turing machine time hierarchy"*. Theoretical Computer Science. Elsevier Science B.V. **26** (3): 327–333.

[15] A. A. Tsay, W. S. Lovejoy, David R. Karger, *Random Sampling in Cut, Flow, and Network Design Problems*, Mathematics of Operations Research, 24(2):383–413, 1999.

[16] Ivanov Viktor V. 2014, *A short proof that NP is not P*. International Journal of Pure and Applied Mathematics IJPAM pp 81-88 .

[17] Woeginger GJ (2016) *The P versus NP page* ,https://www.win.tue.nl/~gwoegi/P-versus-NP.htm

[18] Yannakakis M. 1998 *"Expressing combinatorial optimization problems by linear programs"* Proceedings of STOC 1988, pp. 223-228.