$P \neq NP$, By accepting to make a shift in the Theory (Time as a fuzzy concept), version 3

The Structure of a Theory (TC*, Theory of Computation based on Fuzzy time)

Farzad Didehvar

didehvar@aut.ac.ir

Amir Kabir University of Technology (Tehran Polytechnic)

Abstract. One of the possible hypotheses about time is to consider time as fuzzy concept, in such a way that two instants of time could be overlapped. Historically, some Mathematicians and Philosophers have had similar ideas like Brouwer and Husserl [8].

Throughout this article, the impact of this change on Theory of Computation and Complexity Theory are studied. In order to rebuild Theory of Computation in a more successful and productive approach to solve some major problems in Complexity Theory, the present research is done. This novel theory calls the Theory of TC*.

Keywords. $\mathrm{P} \neq \mathrm{NP}$, P=BPP, Fuzzy Time, Probabilistic Time, TC*, Reducibility, Complexity Theory Problems

1. Introduction

Here, the author tries to build the structure of the Theory of computation based on considering time as a fuzzy concept.

In fact, there are reasons to consider time as a fuzzy concept. In this article, the author doesn't go to this side but note that Brower and Husserl views on the concept of time were similar [8]. Some reasons have been given for it in [3].

Throughout this article, the author presents the Theory of Computation with Fuzzy Time. Given the classic definition of Turing Machine, the concept of Time is modified to Fuzzy time. This new term calls as Theory TC* [2] and this type of computation "Fuzzy time Computation". We have relatively large number of fundamental unsolved problems in Complexity Theory. In the new theory, some of the major obstacles and unsolved problems have been solved [2]. It should be noted that in this article, the writer considers fuzzy number associated to instants of time as a symmetric one. The point about the symmetry is in the proof of Lemma 3, although it is generalizable.

In particular, the new classes of complexity Theory, P*, NP*, BPP* in the TC* analogues to the definitions of P, NP, BPP defines as their natural alternative definition. Here, we will see P*≠ NP*, P*= BPP*. Finally, we have Theorem 4.

2. Reducibility

In this section, we first define the quasi-order relation in TC* analogues with the m-reducibility in TC.

It should be reminded that a fuzzy time Turing Machine is a Turing Machine which works in a fuzzy time.

In addition, here, the Turing Machine has two tuple (M, S). Whereas, M is a Turing machine in the usual sense and S is a polynomial function. Meanwhile, M runs in bounded time by S equivalently, M(x) in less than S([x]) steps is computed.

First, we remind the Classical definition of m-reducibility:

$Y >_m X$, if there is a polynomial time computable function f such that:

$$x \in X \leftrightarrow f(x) \in Y$$

Associated definition in TC*

Definition 1: For $\alpha > \frac{1}{2}$, $Y >_{\boldsymbol{m}}^{\boldsymbol{\alpha}} X$ if there is a polynomial time computable* function f such that:

1. $x \in X \,\&\, f(x) \downarrow \mathit{in\ bounded\ time} \leftrightarrow (f(x) \in Y)$

2. Pr $(f(x) \downarrow \mathit{in\ bounded\ time}) > \alpha$

A Computable* function f is a function that is computable by a fuzzy time Turing machine.

By bounded time, for the function f there exists a Polynomial function h such that $f(x) \downarrow$ in less than h (length(x)) steps.

$Y >_{\boldsymbol{m}}^{\boldsymbol{\alpha}} X$ is represented by a 5-tuple, (Y, X, f, S_f, α), $S_f(x)$ is the number of steps that f(x) is computed. it defines as follows

$$Y >_{\boldsymbol{m}}^{\boldsymbol{\alpha}} X \leftrightarrow (Y, X, f, S_f, \alpha) \text{ is an acceptable 5-tuple}$$

Are these definitions independent from the value of α? $(\alpha > \frac{1}{2})$

In the first step, to answer the above question, the following simple lemma is needed.

Lemma 1. Let for $1 > \alpha > \frac{1}{2}$, (Y, X, f, S_f, α) is an acceptable 5-tuple then for any $1 > \beta > \frac{1}{2}$ there is a computable function g in which (Y, X, g, S_g, β) is a 5-tuple.

Proof. Actually, there is k, so that g= (k times repeating f, till we reach a solution with probability β) . It is easy to understand that such a k exists.

Definition 2. Lemma 1 indicates for $1 > \alpha > \frac{1}{2}$, $Y >_{\boldsymbol{m}}^{\boldsymbol{\alpha}} X$ and it is independent from α. So, we define $Y >'_{\boldsymbol{m}} X$, if for some $\alpha\,(1 > \alpha > \frac{1}{2})$, $Y >_{\boldsymbol{m}}^{\boldsymbol{\alpha}} X$.

Lemma 2. $>'_{\mathbf{m}}$ is a quasi-order relation.

Proof. $\mathrm{X} >_{\mathbf{m}}^{\boldsymbol{\alpha}} \mathrm{Y}$ implies $\forall \frac{1}{2} > \varepsilon > 0\ \ \mathrm{X} >_{\mathbf{m}}^{\mathbf{1-\varepsilon}} \mathrm{Y}$ (*)

$\mathrm{Y} >_{\mathbf{m}}^{\boldsymbol{\alpha}} \mathrm{Z}$ implies $\forall \frac{1}{2} > \varepsilon > 0\ \ \mathrm{Y} >_{\mathbf{m}}^{\mathbf{1-\varepsilon}} \mathrm{Z}$ (**)

From (*), (**) we have $\forall\ \frac{1}{2} > \varepsilon > 0 \quad \mathrm{X} >_{\mathbf{m}}^{(\mathbf{1}-\varepsilon)\hat{}2} \mathrm{Y}$ (***).

Lemma 3. $Y >_m X$ implies $Y >'_{\boldsymbol{m}} X$.

Proof. At this point, let us consider that fuzzy number is symmetric.

We have computable function f such that

$$x \in X \leftrightarrow f(x) \in Y$$

f is supported by (M, S_f). The computation of f on x can be depicted by the following transition of configurations in time $S_f(x)$ to achieve the final configuration.

Now, as mentioned above, the author changes time to be fuzzy. Now the probability of reaching or passing the final configuration is higher than the probability of not reaching this point.

According to the rules of probability and the above comments, the probability of reaching or passing to the final configuration is more than $\frac{3}{4}$ and the probability of not reaching this final configuration is less than $\frac{1}{4}$ if we consummate 2 $S_f(x)$ unit of time. Likewise, by consumption of p $S_f(x)$ unit of time, the probability of reaching to the final configuration or passing it is more than $1 - \frac{1}{p^n}$ and the probability of not reaching this final configuration is less than $\frac{1}{p^n}$. So, we have, $Y >'_{\boldsymbol{m}} X$.

Remark 1. Using lemma 3, suppose we have a computation by Turing Machine (M, S_f) and the input x and classical time. If we change the classical time to symmetric fuzzy time, the probability of reaching the final state is more than $\frac{1}{2}$. . As a conclusion, if we consider for computation $(M, k\,S_f)$, the probability of reaching the final state is more than $1 - \frac{1}{2^k}$.

2.2 P*, NP*, NP*-hard, NP*-Complete

One of the main questions here is how do we define the most important classes of Complexity Theory in the new theory? As a start we try to define P*. As the first attempt, let we try to define it as follows:

P* is the class of all problems that can be determined by a Fuzzy Turing Machine (M, S).

But what exactly do we mean by decidable? Since it is possible that we do not reach to the final state, we should consider about the possibility of x ϵ p for any pϵP* when x belongs to p, and the possibility of x/ϵp when x belongs to p^c. Hence, by above consideration we are able to define P* as follows:

Definition 3: P* is a class of problems for any pϵ P* and the probability α. we have a polynomial $Q\alpha, p$ and an associated algorithm $A\alpha, p$ for solving p by probability α such that $Q\alpha, p$ is upper bound of the computation time.
Equivalently, for any pϵ P* (p as a language) and probability α we have an associated algorithm $B\alpha, p$ and a polynomial $Q\alpha, p$ as an upper bound of the computation time.
xϵp → By probability α, $B\alpha$,p (x)=1
x/ϵp → By probability α, $B\alpha$,$p(x)$=0

This is analogues to the definition of the class BPP.
Equivalently, by considering time as a Fuzzy concept we have BPP.
By above considerations it is easy to see:

Theorem 1 P*=BPP* [1], [2].

The next natural question in TC* is the situation of the problem P vs NP, more exactly P* vs NP*.

Proposition 1 Random Generator exists [1], [2].

Proof. The following algorithm demonstrates that there is a random generator. It is sufficient to consider an algorithm that in interval times [2n,2n+1], it emits as an output 0 and in interval times [2n+1,2n+2], it emits 1, when time is considered as a classical concept. Now by considering time as a fuzzy concept, it is easy to see that we have a random number generator. More exactly, by considering fuzzy instant of time as functions which their domain is T_0 (a copy of R) we have probability function p(x), $1 > p(x) > 0$. Such that for any X+t belongs to T_0, $1 > t > 0$ and X is a natural number
If X is an odd number by probability p(t) in X+t, it emits 1 (p(t) is near to 1)
If X is an even number by probability p(t) in X+t, it emits 0 (p(t) is near to 1)
Clearly, p(x) is a periodic function.

First, let we consider the following definition of NP problems.

Definition 4: The Complexity class **NP** is the set of decision problems like D such that there is deterministic polynomial time Turing machine $\mathrm{M_D}$ and $\mathrm{p_D}, \mathrm{q_D}$ such that for every input x with length $\mathrm{x'}$ ($\mathrm{l(x)=x'}$)

1. x belongs to D implies there exists string z with length $\mathrm{q_D(x')}$ such that for all string y with length $\mathrm{p_D(x')}$ $\mathrm{Pr(M_D(x,y,z) = 1) = 1)}$
2. x belongs to D implies for all string z with length $\mathrm{q_D(x')}$ such that for all string y with length $\mathrm{p_D(x')}$ $\mathrm{Pr(M_D(x,y,z) = 0) = 1}$ (The definition is Quoted [7])

By considering the above definition and by fuzzifying time we have the definition of NP^*.

We define NP*-hard, NP*-Complete likewise in below

Definition 5 X is NP*-hard if for any $Y \in \mathrm{NP}^*$, $X >'_{\boldsymbol{m}} Y$.

Definition 6 X is NP*-Complete if X is NP*-hard and $X \in \mathrm{NP}^*$.

Theorem 2 SAT is NP*-Complete.

Proof. SAT belongs to NP, hence $SAT \in \mathrm{NP}^*$, by definition.

The analogues of the proof of Cook-Levin's theorem by repeating it, and due to the reduction associated with the function f when the time is fuzzy, we have the same function f and considering $>'_{\boldsymbol{m}}$ instead of m-reducibility. Lemma 3 guarantees the proof of the theorem.

In [2], by defining the concepts of P, BPP in the new framework we have $\mathrm{P^*}, \mathrm{BPP^*}$. It is shown that the new classes $\mathrm{P^*}, \mathrm{BPP^*}$ are both equivalent to BPP. In contrast, what is the replacement of the NP class in this new framework? To illustrate NP problems in the Theory of Algorithm, it is required to define a new class for it. Possibly the best choice in probabilistic classes in this purpose is MA [4], [7] (introduced by Laszlo Babai, Shafi Goldwasser, Micheal Sipser).

The MA complexity class is known as a candidate for NP problems in probabilistic classes, we also have a theorem states [5], [6]

$$\mathrm{P = BPP \rightarrow MA = NP}$$

This point besides $\mathrm{P^* = BPP^*}$ strengthen our choice. So, let we define the NP concept in fuzzy time by applying the definition of MA.

Here, we define MA in Two-sided version definition [7].

Definition 7 The Complexity class **MA** is a set of decision problems like D such that there are deterministic polynomial time Turing machine $\mathrm{M_D}$ and $\mathrm{p_D}, \mathrm{q_D}$ such that for every input x with length x' ($\mathrm{l(x)=x'}$)

3. x belongs to D implies there exists string z with length $\mathrm{q_D}(\mathrm{x}')$ such that for all string y with length $\mathrm{p_D}(\mathrm{x}')$ $\mathrm{Pr(M_D(x,y,z) = 1)} \geq {}^2/_3$)
4. x belongs to D implies for all string z with length $\mathrm{q_D}(\mathrm{x}')$ such that for all string y with length $\mathrm{p_D}(\mathrm{x}')$ $\mathrm{Pr(M_D(x,y,z) = 0)} \geq ⅔$ (The definition is Quoted [7])

In conclusion, as the literature on Theory of Computaion changes from classical to fuzzy times,

Complexity Theory classes change to new classes. Likewise, we have new problems in new theory.

The list of new possible classes is:

$\mathrm{P^*, BPP^*}$ and $\mathrm{MA^*, AM^*}$

Instead of $\mathrm{P = NP}$ problem we have the following problems

$$\mathrm{BPP^* = MA^*}$$

$$\mathrm{BPP^* = AM^*}$$

$$\mathrm{MA^* = AM^*}$$

The two last questions remained unproved.

It is easy to see:

1. $\mathrm{P^* = BPP^*}$
2. $\mathrm{NP^* = MA^*}$ (Considering certificate definition of NP)
3. $\mathrm{MA^* = MA}$

Chapter 2. Pseudorandom generator & $\mathbf{NP^+}$

Pseudo-random generators play a major role in Theory of computation. The existence of pseudo-random generator using classical time leads us to P≠NP. What can be done about theory of computation when we consider time as a fuzzy concept ($\mathrm{TC^*}$)?

By proposition 1, more strongly, we have random generator in our Theory,

To obtain our main result in Theorem3, let we define NP+.

Definition 10 (NP+) Non deterministically guess the input for deterministic Turing machine M, let we call this new machine $\mathrm{M+}$.

NP+ are the set of languages which accept by some M+.

When we consider time as a fuzzy concept in above, we have NP+*.

NP+ and NP and NP+* are subsets of NP*.

Theorem 3: P*= NP* & the existence of random generator leads us to a contradiction, moreover by proposition 1 we have P*≠ NP*.

(Hint of proof: P*= NP* implies NP+* is a subset of P*. First, we select all the seeds non deterministically, in a high probability we generate all random numbers. Since P*=NP*, the generator is not random. But by Proposition 1, we have a random generator.)

Corollary. PH* doesn't collapse.

Some Problems in New Theory:

1- Creativity and P vs NP

2-MA*=AM*

3-P*=NP* ∩ CO-NP*

Theorem 4 $\mathrm{P} \neq \mathrm{NP}$.

To prove $\mathrm{P} \neq \mathrm{NP}$, we apply Theorem 2 and lemma 3.

Suppose $\mathrm{P} = \mathrm{NP}$ and we remind that SAT is a NP-Complete problem. Hence, there is an algorithm A which solves SAT in Polynomial time.

Considering Fuzzy time, A also solves SAT in polynomial time, and SAT belongs to P*. SAT is NP*-Complete, so P*=NP*. A contradiction.

Consequently, $\mathrm{P} \neq \mathrm{NP}$.

Conclusion. Here, it is shown that by considering time as a fuzzy concept, the major results in solving some of the famous problems in Complexity Theory are such that they adapt to people's intuitions and expectations in Theory of Algorithm. In brief, P*≠NP*, P*=BPP*. Finally, we have P≠NP.

Reference:

1. F.Didehvar, By considering Fuzzy time, P=BPP (P*=BPP*), Philpaper 2020

2. F.Didehvar, Fuzzy Time & NP Hardness (P*=BPP*, P* $\neq$ NP*) , Philpaper 2020

3. F.Didehvar, Fuzzy time, A Solution of Unexpected Hanging Paradox, Philpapers 2019

4. L.Babai “TRADING Group Theory for Randomness”, STOC’85: Proceedings of the seventeenth annual ACM symposium on Theory of Computing, ACM, pp.421-429, 1985

5. O.Goldreich, In a world of P=BPP

6. O.Goldreich, Studies in Complexity and Cryptography: Miscellanea on the interplay

between Randomness and Computation , Vol 6650 of Lecture Notes in Computer

Science, Springer 2011, P 43.

7. S.Goldwasser; M.Sipser “Private coins versus public coins in interactive proof systems”, STOC’86: Proceedings of the Beighteenth annual ACM symposium on Theory of Computing, ACM, PP.59-68, 1986

8. Van Aten M, On Brouwer, Wadsworth Philosopher’s Series, 2004 (Persian translation by Ardeshir.M)