P ? NP Proof (Millennium Prize Problem Solved using the Proof of X ± Y = B at System 1)

Martins Kolawole Alabi

Received: 8 December 2013 Accepted: 1 January 2014 Published: 15 January 2014

Abstract

This paper is about the proof of P ? NP based on the limiting factor that hinders the proof of P = NP when we consider randomness in the subset sum problem algorithm presented in this paper. Randomness in this paper is referred to an act of inputting integers to the program without pattern. The paper is not about to show case the extent in which the subset problem could be solved but to prove that P = NP does not exist in polynomial time. Literally, polynomial time means that as the complexity of the problem grows, the difficulty in solving it doesn?t grow too fast. The proof of x ± y = b at system 1 is the premise that was used in this paper to prove that P ? NP. The proof of x ± y = b systems are forms of x ± y = b that was derived from the coexistence of three quantities denoted by n, n + 1, n + 2 where n represents any positive integer. See the proof of x ± y = b at references for details. The proof of x ± y = b at system 1 is the proof of a mathematical method that proves something can evolve from nothing and its graph shows that the shape of the universe is a cone and this can further be mapped with an expanding universe or universes to locate the point of the big bang (a hypothetical point in space where the universe began). See No.4, at references for details.

Index terms— input, systems, subset sum problem, algorithm, p ? np, the proof of x ± y = b.

1 I.

Subset Sum Problem onsider the subset sum problem, an example of a problem that is easy to verify, but whose answer may be difficult to compute. Given a set of integers, does some nonempty subset of them sum to 0? For instance, does a subset of the set {-2, -3, 15, 14, 7, -10} add up to 0? The answer "yes, because {-2, -3, -10, 15} adds up to zero" can be quickly verified with three additions.

There is no known algorithm to find a subset that sum to 0 in polynomial time. Thus it takes a very long time to find a subset that sum to 0 as the complexity of the problem grows on a deterministic Turing machine given the computer's present state and any inputs (there is only one possible action that the computer might take) and sequential (it performs actions one after the other). Literally, polynomial time means that as the complexity of the problem grows, the difficulty in solving it doesn't grow too fast. Therefore, if there is a known algorithm to find a subset that sum to 0 in polynomial time then P = NP but if there is no known algorithm to find a subset that sum to 0 in polynomial time and it can be proved then P ? NP.

2 II.

3 Premise

Computers are deterministic and they cannot identify subset that sum to 0 or subset that do not sum Author: e-mail: alabikmartins@gmail.com to 0 without cause (a thing that makes something happen). It can be observed that there is no other information given to the computer rather than the input themselves. Therefore, for us to have a solution to the P vs NP problem the input must equate to something that is provable. Otherwise, the input is meaningless.

The proof of x ± y = b at system 1 is the premise that was used in this paper to prove that P ? NP where b is the total sum of input. The proof of x ± y = b at system 1 gives a solution when integers that sum to 0 combines with one or two integers that do not sum to 0 regardless of input size.

4 III.

5 Assertion

If P = NP. Then a determinant would be found in polynomial time.

Determinant (a factor that causes something) here is referring to an integer n that gives indication or hint about a subset that sum to 0 when the proof of x ± y = b at system 1 is established in the subset sum problem. They can be classified as a cause for subset that sum to the total sum of input.

Given that x ± y = b where b is the solution to any given number x and y for which x is number i or j.

So that, i + y = b. Also that, j -y = b.Let i = (nb + n) / n + 1. Let y = (b -n) / n + 1. Let j = (nb + 2b -n) / n + 1.

Therefore the proof of x ± y = b system is given as:(nb + n) / n + 1 + (b -n) / n + 1 = b. (nb + 2b -n) / n + 1 -(b -n) / n + 1 = b.

System 'n' is a system which x ± y = b belongs for every given positive integer n. For instance, when n = 1.Let i = (b + n) / 2. Let y = (b -n) / 2. Let j = (3b -n) / 2.

Therefore the proof of x ± y = b at system 1 is given as:(b + n) / 2 + (b -n) / 2 = b. (3b -n) / 2 -(b -n) / 2 = b.

6 C

Abstract-The proof of x ± y = b at system 1 is the premise that was used to prove that P ? NP where b is the total sum of input in the subset sum problem. The proof of x ± y = b systems are forms of x ± y = b that was derived from the coexistence of three quantities denoted by n, n + 1, n + 2 where n represents any positive integer. The proof of x ± y = b at system 1 is a computable function definable by an algorithm where n is the argument (input value) of the function to the corresponding output value b. System 1 can be defined as a system which the proof of x ± y = b belongs when n = 1. The proof of x ± y = b at system 1 is the proof of a mathematical method that proves something can evolve from nothing and its graph shows that the shape of the universe is a cone and this can further be mapped with an expanding universe or universes to locate the point of the big bang .i.e. a hypothetical point in space where the universe began. See the reference list for details.

7 Search for Determinant

Given a set of integers {-2, -3, 15, 14, 7, -10} does some nonempty subset of them sum to 0? Let i denote each integer in the set.

Since input {-2, -3, 15, 14, 7, -10} equals 21. Then b = 21.

The proof of x ± y = b at system 1 is implicit. When we know i and b we can know n by making use of the formula n = (2 × i) -b derived from the proof of x ± y = b at system 1. For i = -2, n = -25.

For i = 15, n = 9. For i = 14, n = 7. For i = 7, n = -7.

For i = -3, n = -31. For i = -10, n = -41. i = 14 when n = 7 and i = 7 when n = -7.

Since 7 + (-7) = 0. Then n = 7 is a determinant and n = -7 is a determinant.

First hint: If the sum of equal and opposite values for n equals 0 outputs all integers in the set except integers that reference equal and opposite values for n. Therefore the answer is subset {-2, -3, 15,-10} which adds up to zero.

Given a set of integers {-2, -3, 15, 14, -10} does some nonempty subset of them sum to 0? Let i denote each integer in the set.

Since input {-2, -3, 15, 14, 7, -10} equals 14.Then b = 14. By using the formula n = (2 × i) -b derived from the proof of x ± y = b at system 1. For i = -2, n = -18. For i = -3, n = -20.

For i = 15, n = 16. For i = 14, n = 14.

For i = -10, n = -34. i = 14 when n = 14. Since n = 14. Then 14 is a determinant.

Second hint: If n is equal to the total sum of input output all integers in the set except the integer that has equal value with the total sum of input. Therefore the answer is subset {-2, -3, 15,-10} which adds up to zero.

V.

8 Complexity

The complexity of the problem grows as we increase the length of random input. For instance, given a set of integers {-2, -3, 15, 14, 7, -10, 50} does some nonempty subset of them sum to 0? Let i denote each integer in the set.

Since input {-2, -3, 15, 14, 7, -10, 50} equals 71.Then b = 71.

By using the formula n = (2 × i) -b derived from the proof of x ± y = b at system 1.

For i = -2, n = -75. For i = -3, n = -77.

For i = 15, n = -41. For i = 14, n = -43. For i = 7, n = -57.

For i = -10, n = -91. For i = 50, n = 29.

No answer could be found because of absence of a determinant. Therefore no hint is applicable.

9 a) Comment

The algorithm presented in this paper outputs subset that sum to 0 quickly if we enter integers that sum to 0 as many as possible with one or two integers that do not sum to 0 via the program below e. Is P = NP? This is the real question we are concerned with. Is the set of problems in P actually the same as the set of problems in NP? The program is a P algorithm; an answer in polynomial time is called P. Questions for which an answer can be verified in Thus it is concluded that the existence of an algorithm exists in polynomial time (P) only if the algorithm runs in polynomial time regardless of input size. ? Here's a P program that does solve the problem ? It does solve the problem because the problem does not exist in P.

? Therefore P ? NP.

polynomial time is called NP. The program terminates and those NP-complete problems do not exist in P therefore, P ? NP. [1]

Figure 1: I

.1 Conclusion

Subset sum problem is NP-complete and the proof of x ± y = b at system 1 has been established in the subset sum problem and each number is equally probable. This justifies that each integer in the set of integers are in conjunction with 'b'. Therefore, the algorithm and the assertion presented in this paper can be dependent on as reliable.

The program outputs subset that sum to 0 quickly when integers that sum to 0 combines with one or two integers that do not sum to 0 regardless of input size. As the complexity of the problem grows the program terminates quickly without an answer.

[Martins Kolawole] *Artificial Intelligence AI Choice Mechanism Hypothesis of a Mathematical Method that Describes Quantum Numbers, Quantum Entanglement And Expanding Universe or Universes Proving That Something can evolve from Nothing (.i.e. numbers evolving from non-existent illusion) Advances in Physics Theories and Applications*, Alabi Martins Kolawole . IISTE.www.iiste.org/Journals/index.-php/APTA/article/view/2155/2165

[Martins Kolawole Alabi et al.] *Mathematical Theory and Modeling*, Martins Kolawole Alabi , The , Of X ± Y = B . IISTE.www.-iiste.org/Journals/index.php/MTM/article/view/10520/13697

[P versus NP problem-Wikipedia] *P versus NP problem-Wikipedia*,

[Subset sum problem-Wikipedia] *Subset sum problem-Wikipedia*,