An Study on NP-completeness in Fuzzy Context

Mathews M. George
B.A.M College, Thuruthicad
Pathanamthitta, Kerala-689 597, INDIA

Abstract

The original (0 or 1) set theory models the world as black or white and makes no provision for sets of grey. Life is not always black and white. This two valued logic has proved effective and successful in solving well defined problems. However, a class of problems exists which are typically complex in nature, and are often left to human beings to deal with. It is known that no polynomial algorithm could be found for the problems in NP or NP-hard problems like travelling salesman Problem, the Hamiltonian Cycle Problem. While a method for computing the solutions to NP-complete problems using a reasonable amount of time remains undiscovered, computer scientists and programmers still frequently encounter NP-complete problems. NP-complete problems are often addressed by using algorithms. The objective of this paper is to introduce and study more about NP- Completeness in the fuzzy context.

Key words: Algorithm, fuzzy sets, fuzzy operations, fuzzy algorithm, complexity, the classes P and NP, NP-completeness, NP-hardness**.**

1. INTRODUCTION

The origin of the word 'algorithm' dates back to Euclid who obtained a step-by-step procedure to find the gcd of two integers. During the time of Leibnitz (1646 – 1716) and perhaps earlier, there were attempts to provide algorithmic proofs for problems involving numerical computation. David Hilbert's (1862-1943) aim was to device a formal mathematical system in which all problems can be precisely formulated in terms of statements which are either true or false. If Hilbert's aim was fulfilled then any problem which was well defined could be solved by executing the algorithm. In 1931 Kurt Gödel discovered that no algorithm of the type desired by Hilbert exists. Gödel's work is the famous *incompleteness theorem.*

Gödel's incompleteness theorem states that there is no algorithm whose input can be any statement involving integers and whose output tells whether or not the statement is true. This was endorsed by A. Church, S. Kleene, E. Post and many others of his time. Gödel defined an algorithm as a sequence of rules for forming complicated mathematical functions out of simpler mathematical functions. Church used a formalism called the lambda calculus, while Turing used a hypothetical machine called a *Turing machine*. The Church Turing thesis states that we can give a reasonably good definition for the word algorithm. In modern terminology, we can define an algorithm as a set of instructions which can be executed on a computer.

In its four and a half decades of existence since the seminal paper of L.A. Zadeh, fuzzy logic has absolved the stages of wide theoretical research, hardware-supported fuzzy systems and industrial applications. Basic for the study of fuzzy computability is the concept of fuzzy algorithm that was introduced by Zadeh in 1968, but not further developed. As in the case of crisp computability, alternative to fuzzy algorithms, fuzzy Turing machines may be considered.

2. PRELIMINARIES

2.1 Complexity

It is the computer science field called complexity theory which asks and attempts to answer questions about the amount of use of computational resources *time, memory and hardware*. The complexity of a problem is just the complexity of the best algorithm which solves that problem. We consider *time* and memory as resources and the complexity class is formed by placing a bound (a function of the input value) on the amount of a particular resource that an algorithm may use. That is, a function is in a given class if there is an algorithm for computing the function, in which the amount of resource does not exceed the bound.

The amount of any resource used by an algorithm may vary with size of the input data. We are interested only in finding the most 'efficient' algorithm for solving a problem. The time requirements of an algorithm are conveniently expressed in terms of a single variable, the size of a problem. That is, the amount of input data needed to describe the instance.

2.2 Polynomial time Algorithm

In producing a complete analysis of the computing time of the algorithm we have two phases:

(i) A *priori* analysis

(ii) A *posteriori* analysis

In a *priori* analysis we obtain a function which bounds the algorithm's computing time. In a *posteriori* analysis we collect actual statistics about the algorithm's consumption time and space while it is executing.

2.3 The O – Notation

There are different types of mathematical notations which are very useful for the kind of analysis. One of these is O-notation.

Definition 2.3.1 A function f(n) is said to be O(g(n)) if there exists a constant c such that $|f(n)| \leq c\,|g(n)|$ for all $n \geq n_0$.

Suppose we are determining the computing time, f(n) , of some algorithm. The variable n might be the number of inputs and outputs, their sum or the magnitude of one of them.If $P(n) = a_m n^m + \ldots + a_1 n + a_0$ is a polynomial of degree m then $P(n) = O(n^m)$.That is, if we can describe the frequency of execution of a statement in an algorithm by a polynomial such as P (n), then that statement's computing time is $O(n^m)$.

The amount of resource used as a function of n and as n grows larger; some term in the function may dominate the other terms. The dominating term is called *asymptotic* behavior of the algorithm. Algorithm whose asymptotic behavior is 2^n or more generally c^n for some constant c, are called *exponential* algorithms. The algorithm whose asymptotic behavior is n, n^2 or more generally n^k for some constant k, are called *polynomial* algorithm.

Definition 2.3.2 The *time complexity function* of an algorithm expresses its time requirements by giving, for each possible input length, the largest amount of time needed by the algorithm to solve a problem. An algorithm is said to have time complexity O(f(n)) if the number of steps needed to process data of size n is at most c f(n), where f(n)is some function of n and c is a constant.

Definition 2.3.3 A *polynomial time algorithm* is defined to be one whose time complexity function is O(p(n)) for some polynomial function p, where n is the input length. An algorithm whose time complexity function cannot be so bounded is called an exponential *time algorithm*. Polynomial time algorithm is generally regarded as being much more desirable than exponential time algorithm. That is, a problem has not been 'well solved' until a polynomial time algorithm is known for it. A problem is so hard that no polynomial time algorithm can possibly solve it.

Definition 2.3.4 Any problem for which the answer is either zero or one is called a *decision problem*. An algorithm for a decision problem is termed a *decision algorithm*. Any problem that involves the identification of an optimal solution (either maximum or minimum) value of a given cost function is known as an *optimization problem*. An *optimization algorithm* is used to solve an optimization problem.

Definition 2.3.5 A decision problem is in P if there is a known polynomial-time algorithm to get that answer. A decision problem is in NP if there is a known polynomial-time algorithm for a non-deterministic machine to get the answer.

Result 2.3.6 Problems known to be in P are trivially in NP - the nondeterministic machine just never troubles itself to fork another process, and acts just like a deterministic one. There are problems that are known to be neither in P nor NP; a simple example is to enumerate all the bit vectors of length n. No matter what, that takes 2^n steps.

3. NP-COMPLETENESS

The concept of NP-complete was introduced by Stephen Cook (1971) in a paper entitled the complexity of theorem-proving procedures on pages 151–158 of the Proceedings of the 3rd Annual ACM Symposium on Theory of Computing. Since Cook's original results, thousands of other problems have been shown to be NP-complete by reductions from other problems previously shown to be NP-complete In this section we discuss the Theory of NP-Hard and NP-Complete problems.

A problem is NP-Complete if it has the property that it can be solved in polynomial time if and only if all other NP-Complete problems can also be solved in polynomial time. If an NP-Hard problem can be solved in polynomial time, then all NP-Complete problems can be solved in polynomial time. All NP-Complete problems are NP-Hard, but not all NP-Hard problems are NP-Complete.

3.1 Nondeterministic Algorithms

Algorithms such that the result of every operation is uniquely defined are called deterministic algorithms. A machine capable of executing a deterministic algorithm is called a deterministic machine.

Algorithms such that the result of every operation is not uniquely defined are called non deterministic algorithms. A machine capable of executing a nondeterministic algorithm is called a nondeterministic machine.

Non deterministic fuzzy machine, as defined, do not exist in practice.

NP-complete is a subset of NP, the set of all decision problems whose solutions can be verified in polynomial time; *NP* may be equivalently defined as the set of decision problems that can be solved in polynomial time on a nondeterministic Turing machine. A problem *p* in NP is also in NPC if and only if every other problem in NP can be transformed into *p* in polynomial time. NP-complete can also be used as an adjective: problems in the class NP-complete are known as NP-complete problems.

3.2 Definition of NP-completeness

A decision problem L is NP-complete if:

1. $L \in NP$, and
2. Every problem in NP is reducible to L in polynomial time.

L can be shown to be in NP by demonstrating that a candidate solution to L can be verified in polynomial time.

Note that a problem satisfying condition 2 is said to be NP-hard, whether or not it satisfies condition 1.

A decision problem *L* is *NP-complete* if it is in the set of NP problems so that any given solution to the decision problem can be verified in polynomial time, and also in the set of NP-hard problems so that any NP problem can be converted into *L* by a transformation of the inputs in polynomial time.

Although any given solution to such a problem can be verified quickly, there is no known efficient way to locate a solution in the first place; indeed, the most notable characteristic of NP-complete problems is that no fast solution to them is known. That is, the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows. As a consequence, determining whether or not it is possible to solve these problems quickly, called the P versus NP problem, is one of the principal unsolved problems in computer science today.

NP-complete problems are studied because the ability to quickly verify solutions to a problem (NP) seems to correlate with the ability to quickly solve that problem (P). It is not known whether every problem in NP can be quickly solved—this is called the P = NP problem. But if any single problem in NP-complete can be solved quickly, then every problem in NP can also be quickly solved, because the definition of an NP-complete problem states that every problem in NP must be quickly reducible to every problem in NP-complete (that is, it can be reduced in polynomial time). Because of this, it is often said that the NP-complete problems are harder or more difficult than NP problems in general.

Definition 3.3 A problem L′ is reducible to L if there is a polynomial-time many-one reduction, a deterministic algorithm which transforms any instance $l' \in L'$ into an instance, $l \in L$ such that the answer to l is *yes* if and only if the answer to l′ is *yes*. To prove that an NP problem L is in fact an NP-complete problem it is sufficient to show that an already known NP-complete problem reduces to L.

A consequence of this definition is that if we had a polynomial time algorithm for L, we could solve all problems in NP in polynomial time. Some NP-complete problems, indicating the reductions typically used to prove their NP-completeness

Thus, the easiest way to prove that some new problem is NP-complete is first to prove that it is in NP, and then to reduce some known NP-complete problem to it. Therefore, it is useful to know a variety of NP-complete problems.

4 INTRODUCTIONS TO FUZZY SETS

The original (0 or 1) set theory was invented by the nineteenth century German mathematician George Kantor. But this two-valued (Crisp) set has some shortcomings as the logic it is based on. The first logic of vagueness was introduced in 1920 by the Polish Philosopher Jan Lukasiewicz. He developed sets with possible membership values of 0, 1/2, and 1, later extending it by allowing an infinite number of values between 0 and 1. Later in twentieth century, the nature of mathematics, real life events, and complexity all played roles in the examination of crispness. So did the discovery of physicists such as Albert Einstein (relativity) and Werner Heisenberg (uncertainty). Einstein was quoted as saying, "As far as the laws of mathematics refer to reality, they are not certain, as far as they are certain, they do not refer to reality."

The next development was in 1937, at Cornell University, where Max Black measured membership in degrees of usage and advocated a general vagueness.

The work of these nineteenth and twentieth century thinkers provided the grist of the mental mill of the founder of fuzzy logic, an American named Lofti Zadeh. In 1960s, he invented fuzzy logic, which combines the concepts of crisp logic and Lukasiewicz sets by defining graded membership. Fuzzy sets were originally introduced in 1965 by Zadeh. Fuzzy systems can be used for estimating, decision-making, and mechanical control systems such as air conditioning, automobile controls and a host of other applications. The British engineer Ebrahim Mamdani was the first to use fuzzy sets in a practical control system. Japan is the world's leading producer of fuzzy based commercial applications. Japanese scientists and engineers were among the earliest supporters of Zadeh's work and had introduced fuzziness in that country. In addition, research on fuzzy concepts and products is enthusiastically pursued in China. There are more fuzzy-oriented scientists and engineers there than in any other country. Thus, fuzzy logic systems have given rise to new technologies applied in different human endeavours.

Since the concept of fuzzy set was introduced by Zadeh in a paper published in 1965, the fuzzy concept was invaded almost all branches of mathematics. The fuzzy topological space was introduced by Chang and since then various notions in classical topology have been extended. A number of research works have been dedicated on development of various aspects of the theory and applications of fuzzy sets. The fuzzy topological space was introduced and The original (0 or 1) set theory was invented by the nineteenth century German mathematician George Kantor. But this two-valued (Crisp) set has some shortcomings as the logic it is based on. The first logic of vagueness was introduced in 1920 by the Polish Philosopher Jan Lukasiewicz. He developed sets with possible membership values of 0, 1/2, and 1, later extending it by allowing an infinite number of values between 0 and 1. Later in twentieth century, the nature of mathematics, real life events, and complexity all played roles in the examination of crispness. So did the discovery of physicists such as Albert Einstein (relativity) and Werner Heisenberg (uncertainty). Einstein was quoted as saying, "As far as the laws of mathematics refer to reality, they are not certain, as far as they are certain, they do not refer to reality." The next development was in 1937, at Cornell University, where Max Black measured membership in degrees of usage and advocated a general vagueness.

The work of these nineteenth and twentieth century thinkers provided the grist of the mental mill of the founder of fuzzy logic, an American named Lofti Zadeh. In 1960s, he invented fuzzy logic, which combines the concepts of crisp logic and Lukasiewicz sets by defining graded membership. Fuzzy sets were originally introduced in 1965 by Zadeh. Fuzzy systems can be used for estimating, decision-making, and mechanical control systems such as air conditioning, automobile controls and a host of other applications. The British engineer Ebrahim Mamdani was the first to use fuzzy sets in a practical control system. Japan is the world's leading producer of fuzzy based commercial applications. Japanese scientists and engineers were among the earliest supporters of Zadeh's work and had introduced fuzziness in that country. In addition, research on fuzzy concepts and products is enthusiastically pursued in China. There are more fuzzy-oriented scientists and engineers there than in any other

country. Thus, fuzzy logic systems have given rise to new technologies applied in different human endeavours.

Since the concept of fuzzy set was introduced by Zadeh in a paper published in 1965, the fuzzy concept was invaded almost all branches of mathematics. The fuzzy topological space was introduced by Chang and since then various notions in classical topology have been extended. A number of research works have been dedicated on development of various aspects of the theory and applications of fuzzy sets. The fuzzy topological space was introduced and

Let X be a non empty set and I = [0, 1]

Definition 4.1 A fuzzy sub set A of X is defined to be a mapping $A : X \to [0,1]$ (or $\mu_A : X \to [0,1]$). The class A is characterized by the function and has the following representation,

$A = \{(x, \mu_A (x)): x \in X\}$.

An ordinary set $A \subseteq X$ is identified with its characteristic function $\chi_A: X \to \{0, 1\}$.

Example 4.2 Let X = {a, b, c, d}. Define $\mu : X \to [0, 1]$ by $\mu(a) = 0.1$, $\mu(b) = 1$, $\mu(c) = 0.4$, $\mu(d) = 0.6$. Then the class A = {(a, 0.1), (b, 1), (c, 0.4), (d, 0.6)} is a fuzzy subset of X.

The fuzzy sets $\underline{0}$ and $\underline{1}$ are given by $\underline{0}(x) = 0$ and $\underline{1}(x) = 1$ for all $x \in X$.

Let I(X) be the family of all the fuzzy sets in X called fuzzy space and the power set P(X) be the class of fuzzy sets whose membership functions have all their values in {0,1}. X is called the carrier domain of each fuzzy subset in it, and I is called the value domain of each fuzzy subset of X.

Definition 4.3 Let μ and σ be any two fuzzy sets in a set X. Then μ is said to be contained in σ , denoted by $\mu \leq \sigma$, if $\mu(x) \leq \sigma(x)$ for all $x \in X$.

If $\mu(x) = \sigma (x)$ for all $x \in X$, μ and σ are said to be equal and we write $\mu = \sigma$.

Definition 4.4 A fuzzy set in X is called a fuzzy point iff it takes the value 0 for all $y \in X$ except one, say, $x \in X$. If its value at x is λ for $\lambda \in (0, 1]$, we denote this point by x_λ where the point x is called its support.

Definition4.5 The fuzzy point x_λ is said to be contained in a fuzzy setμ, denoted by $x_\lambda \in \mu$ if and only if $\lambda \leq \mu (x)$.

In I(X) the following additive operations can be introduced:

Definition 4.6 [1]

i. The sum of two fuzzy sets μ and σ in a set X ,denoted by $\mu \oplus \sigma$,is a fuzzy set in X defined by $(\mu \oplus \sigma)(x) = \min (1, \mu (x) + \sigma (x))$ for all $x \in X$

ii. The difference of two fuzzy sets μ and σ in X ,denoted by $\mu \Theta \sigma$ is a fuzzy set in X defined by $(\mu\Theta\sigma)(x) = \max(0, \mu(x) - \sigma(x))$ for all $x \in X$

iii. The conjunction of two fuzzy sets μ and σ ,denoted by $\mu\& \sigma$, is a fuzzy set in X defined by $(\mu\&\sigma)(x) = \max(0, \mu(x)+ \sigma(x)-1)$ for all $x \in X$

iv. The product $\mu . \sigma$ is a fuzzy set defined by
$(\mu.\sigma)(x) = \mu(x). \sigma(x)$ for all $x \in X$.

In his classical paper [9], Zadeh have introduced the union, the intersection, the complementary and the inclusion as follows:

i. The union $\mu \cup \sigma$ is the fuzzy set in X defined by
$(\mu \cup \sigma)(x) = \max(\mu(x), \sigma(x))$ for all $x \in X$

ii. The intersection $\mu \cap \sigma$ is the fuzzy set defined by
$(\mu \cap\sigma)(x) = \min(\mu(x), \sigma(x))$ for all $x \in X$

iii. The complement of a fuzzy set μ is in a set X, denoted by μ^c is the fuzzy set of X defined by $\mu^c(x) = 1 - \mu(x)$ for all $x \in X$.

iv. The inclusion of fuzzy sets is given by $\mu \subseteq \sigma$ iff $\mu(x) \le \sigma(x)$ for all $x \in X$.

5 FUZZY ALGORITHMS

The concept of fuzzy algorithm may be viewed as a generalization, through the process of fuzzification, of the conventional (non fuzzy) concept of an algorithm. A fuzzy algorithm may contain fuzzy statements containing names of fuzzy sets.

Illustrations [10] Fuzzy algorithm may contain fuzzy instructions such as:

(a) "Set y approximately equal to 10 if x is y approximately equal to 5"
(b) "If x is large, increase y by several units"
(c) "If x is large, increase y by several units; if x is small, decrease y by several units; otherwise keep y unchanged"

The sources of fuzziness in these instructions are fuzzy sets which are identified by their underlined names.

Fuzzy instruction which is a part of a fuzzy algorithm can be assigned a precise meaning by making use of the concept of the membership function of a fuzzy set.

Definition 5.1 Fuzzy algorithms such that the result of every operation is uniquely defined are called deterministic fuzzy algorithms. A machine capable of executing a deterministic fuzzy algorithm is called a deterministic fuzzy machine.

Definition 5.2 Fuzzy algorithms such that the result of every operation is uniquely defined are called deterministic fuzzy algorithms. A machine capable of executing a nondeterministic fuzzy algorithm in this way is called a nondeterministic fuzzy machine.

NP-complete is a subset of NP, the set of all decision problems whose solutions can be verified in polynomial time; *NP* may be equivalently defined as the set of decision problems that can be solved in polynomial time on a nondeterministic fuzzy Turing machine.

Different formulations of fuzzy algorithms are developed. Turing algorithm and Markov normal algorithm are the most important among them. It is shown that the two formulations are equivalent in a certain sense.

In 1920s logicians began to investigate the concept of computability more seriously and buy a kind of miracle, as Godel (1946) later expressed it, computability turned out to be a mathematically precise notion. The notion was first formulated by Turing (1936) and Post (1936), who arrived independently at a definition of computing machine, now called a Turing machine. Just as the notion of a non fuzzy algorithm can be defined precisely in the context of countable sets by placing it in one- one correspondence with a Turing machine, so can the notion of a fuzzy algorithm be given a precise, although restricted, meaning by placing it in one-one correspondence with a Turing machine.

The class fuzzy P is defined to be the class of all decision problems that can be solved in polynomial time by a non-deterministic fuzzy algorithm. The class fuzzy NP is defined to be the class of all decision problems that can be solved in polynomial time by a non-deterministic fuzzy algorithm.

5.3 Fuzzy NP-completeness

A decision problem L is fuzzy NP-complete if:

1. L $\in$ fuzzy NP, and
2. Every problem in fuzzy NP is reducible to L in polynomial time.

L can be shown to be in NP by showing that L can be verified in polynomial time.

Note that a problem satisfying condition 2 is said to be NP-hard, whether or not it satisfies condition 1.

There is often only a small difference between a problem in fuzzy P and a fuzzy NP-complete problem.

Future Studies NP-Complete problems are in NP, and also are NP-Hard, but not all NP-Hard problems are in NP in the fuzzy context.

References

1. D. Butnariu , *Additive Fuzzy Measures and Integrals I,* Journal of Mathematical analysis and applications 93,(1983) 436-452
2. Cook, S.A. (1971). "The complexity of theorem proving procedures". *Proceedings, Third Annual ACM Symposium on the Theory of Computing, ACM, New York*. pp. 151–158.
3. Garey, M.R.; Johnson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W.H. Freeman. ISBN 0-7167-1045-5. This book is a classic, developing the theory, and then cataloguing *many* NP-Complete problems.
4. Mathews M. George, *Stephan Cook's theorem and NP-Completeness,* M. Phil dissertation, Calicut University, Kerala India 1992
5. Michael E. Albertson, Joan P. Hutchinson, *Discrete Mathematics with Algorithms*, John Wiley & Sons, 2003.
6. L.A. Zadeh, *Fuzzy sets, Information and Control* 8 (1965) , 338-352
7. L.A. Zadeh, *Fuzzzy Algorithms*, Information and Control,12,91-102 (1968)
8. H.Z. Zimmermann, *Fuzzy Set Theory and Applications*, Allied Publishers Limited, New Delhi 1996