

In Its Usual Formulation, Fuzzy Computation Is, In General, NP-Hard, But a More Realistic Formulation Can Make It Feasible

1st Martine Ceberio
Department of Computer Science
University of Texas at El Paso
 El Paso, Texas 79968, USA
 mceberio@utep.edu

2nd Olga Kosheleva
Department of Teacher Education
University of Texas at El Paso
 El Paso, Texas 79968, USA
 olgak@utep.edu

3rd Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso
 El Paso, Texas 79968, USA
 vladik@utep.edu

4th Luc Longpré
Department of Computer Science
University of Texas at El Paso
 El Paso, Texas 79968, USA
 longpre@utep.edu

Abstract—The need for most computations comes from the fact that in many practical situations, we cannot directly measure or estimate the desired quantity y – e.g., we cannot directly measure the distance to a star or the next week’s temperature. To provide the desired estimate, we measure or estimate easier-to-measure quantities x_1, \dots, x_n which are related to y , and then use the known relation $y = f(x_1, \dots, x_n)$ to transform our estimates \tilde{x}_i for x_i into an estimate $\tilde{y} = f(\tilde{x}_1, \dots, \tilde{x}_n)$ for y . In situations when x_i are known with fuzzy uncertainty, we thus need *fuzzy computation*. Zadeh’s extension principle provides us with formulas for fuzzy computation. The challenge is that the resulting computational problem is NP-hard – which means that, unless $P=NP$ (which most computer scientists consider to be impossible), it is not possible to solve all fuzzy computation problems in feasible time. To overcome this challenge, we propose a more realistic formalization of fuzzy computation – in which instead of an un-realistic requirement that the corresponding properties hold for all x_i , we only require that they hold for almost all x_i – in some reasonable sense. We show that under this modification, the problem of fuzzy computation becomes computationally feasible.

Index Terms—fuzzy computation, Zadeh extension principle, interval computation, feasible, NP-hard, Monte-Carlo techniques

I. FORMULATION OF THE PROBLEM

Need for computations. To explain our problem, let us recall why we need computations in the first place. The main objectives of science and engineering are:

- to describe the world,
- to predict what will happen in the future, and,
- if necessary, to come up with recommendation of what to do to make the future state of the world better.

This work was supported in part by the National Science Foundation grant HRD-1242122 (Cyber-ShARE Center of Excellence).

The physical world is usually described by the values of the corresponding physical quantities. Thus, to describe the current state of the world, we need to describe the numerical values of all these quantities.

Some of these values we can direct measure or estimate. For example:

- we can directly measure the width of a room;
- by touching a baby’s forehead, we can directly estimate the baby’s body temperature, etc.

However, there are many other quantities which are difficult to measure or estimate directly. For example, it is not easy to directly measure or estimate:

- the distance to a faraway star, or
- the temperature inside the car engine.

This impossibility is even more evident if we are interested in the future values of the quantities of interest: there is no way that we can measure them now, and it is rarely possible to directly estimate these future values – otherwise, we would not need high-performance computers to predict next week’s weather.

If we cannot directly measure or estimate the value of the desired quantity y , a natural idea is to estimate this value indirectly. Namely:

- we find some easier-to-estimate auxiliary quantities x_1, \dots, x_n which are related to the desired quantity y by a known dependence $y = f(x)$, where we denoted

$$x \stackrel{\text{def}}{=} (x_1, \dots, x_n);$$

- then, we measure or estimate these auxiliary quantities;
- finally, we use the resulting estimates \tilde{x}_i to compute the estimate $\tilde{y} = f(\tilde{x}_1, \dots, \tilde{x}_n)$ for the desired quantity y .

For example, to predict the temperature y in El Paso in a week, we can:

- measure the values x_1, \dots, x_n describing the temperature, humidity, and wind speed measurements in a wide area, and then
- use the algorithm $y = f(x_1, \dots, x_n)$ for solving the corresponding partial differential equations to predict how the next-week temperature in El Paso.

Such estimations are the main reason why computations are needed in the first place.

Need for fuzzy computation. In many situations, the values x_i are estimated by experts – or measured by measuring instruments whose uncertainty is described by experts. Experts usually describe the accuracy of their estimates not in precise mathematical terms, but rather by using imprecise (“fuzzy”) words from natural language. For example, an expert can say that the value of x_i is “close to 1”, “the difference between the actual value and 1 is small”, or, more accurately, “close to 1 with an estimation error around 0.1”. To use these imprecise estimates, we need to translate them into precise computer-understandable terms.

For this translation, it is reasonable to use fuzzy logic – a technique specifically designed by Lotfi Zadeh for the purpose of such a translation; see, e.g., [1], [2], [4], [6], [8], [15]. In fuzzy logic, each imprecise property (like “small”) is described by assigning, to each possible value of the corresponding quantity q , the degree $\mu(q) \in [0, 1]$ to which this value satisfies the corresponding property – e.g., is small. The corresponding function $q \rightarrow \mu(q)$ is known as the *membership function*.

After this translation, we have n membership functions $\mu_1(x_1), \dots, \mu_n(x_n)$. Based on these functions, we would like to come up with the membership function $\mu(y)$ describing possible values of $y = f(x_1, \dots, x_n)$. Computing this function is a fuzzy case of the general computation problem; we will therefore call computing $\mu(y)$ *fuzzy computing*.

Traditional formulas for fuzzy computing: Zadeh’s extension principle. How can we compute $\mu(y)$? The main idea behind this computation was proposed by Zadeh himself. This idea is based on a fact that the number Y is a possible value of the quantity $y = f(x_1, \dots, x_n)$ if $Y = f(X_1, \dots, X_n)$ for some possible values X_i of x_i . In other words, Y is possible if:

- either X_1 is a possible value of x_1 and X_2 is a possible value of x_2 , and ..., for some tuple (X_1, \dots, X_n) for which $Y = f(X_1, \dots, X_n)$,
- or X'_1 is a possible value of x_1 and X'_2 is a possible value of x_2 , and ..., for some tuple (X'_1, \dots, X'_n) for which $Y = f(X'_1, \dots, X'_n)$,
- etc.

For each i and for each value X_i , we know the degree to which X_i is a possible value of x_i – this value is $\mu_i(X_i)$, by definition of the membership function. So, to find the expert’s degree of confidence in the above composite statement, we need to be able, in general, to combine the degrees of confi-

dence in two statements S and S' into a degree of confidence in a propositional combination “ S and S' ” or “ S or S' ”.

In fuzzy logic, such a combination is provided by, correspondingly, “and”-operations (also known as *t-norms*) and “or”-operations (also known as *t-conorms*). The simplest such operations are $\min(a, b)$ and $\max(a, b)$. By using these operations, we get the following formula for the desired degree $\mu(Y)$:

$$\mu(Y) = \max_{X_1, \dots, X_n: Y=f(X_1, \dots, X_n)} \min \{\mu_1(X_1), \mu_2(X_2), \dots\}.$$

This formula, first proposed by Zadeh, is known as *Zadeh’s extension principle*, since it extends the function $f(x_1, \dots, x_n)$ used in traditional numerical computations to the case when inputs are fuzzy; see, e.g., [1], [2], [4], [6], [8], [15].

Challenge: in its usual formulation, fuzzy computation is, in general, NP-hard. We have shown that fuzzy computation is practically important. How difficult is it to perform the corresponding computations?

As we will show, in general, fuzzy computations are NP-hard; see, e.g., [3], [14]. Crudely speaking, this implies that, if (as most computer scientists believe) P is different from NP, no algorithm is possible that would solve all fuzzy computation problems in feasible time [3], [9]. We will show that this is even true if we restrict ourselves to quadratic functions

$$f(x_1, \dots, x_n) = a_0 + \sum_{i=1}^n a_i \cdot x_i + \sum_{i=1}^n \sum_{j=1}^n a_{ij} \cdot x_i \cdot x_j.$$

For some inputs, the computation time grows exponentially with the number of inputs n , and thus, already for values $n \approx 50$, becomes larger than the lifetime of the Universe – i.e., completely unrealistic and infeasible.

In practical problems, the number of inputs can be huge. For example, when we estimate the quality of a road, we use the opinion of multiple pavement experts who provide estimates of different quality aspects of different road segments; see, e.g., [10]–[12].

The proof of NP-hardness is very straightforward. Namely, it is well known – and relatively easy to prove – that for the membership function $\mu(y)$, the corresponding α -cut

$$y(\alpha) \stackrel{\text{def}}{=} \{y : \mu(y) \geq \alpha\}$$

is equal to the range of $f(x_1, \dots, x_n)$ when each x_i is in the corresponding α -cut $x_i(\alpha) = \{x_i : \mu_i(x_i) \geq \alpha\}$:

$$y(\alpha) = \{f(x_1, \dots, x_n) : x_i \in x_i(\alpha) \text{ for all } i\}.$$

It is known that even in the simplest case, when the sets $x_i(\alpha)$ are intervals, computing the range is NP-hard for quadratic functions $f(x_1, \dots, x_n)$ [3], [14].

Let us describe this result in precise terms. For simplicity, let us consider the usual case when all α -cuts are intervals, i.e., if $x_i(\alpha) = [\underline{x}_i(\alpha), \bar{x}_i(\alpha)]$. The bounds $\underline{x}_i(\alpha)$ and $\bar{x}_i(\alpha)$ of these intervals must be given, i.e., must be represented in a computer – thus, we can safely assume that these bounds are rational numbers.

The function $f(x_1, \dots, x_n)$ is continuous – as we have mentioned earlier, it is sufficient to consider quadratic functions. Similarly to the bounds of the x -intervals, we can safely assume that the coefficients of this quadratic functions are rational numbers.

Our goal is then to compute the range $y(\alpha) = [\underline{y}(\alpha), \bar{y}(\alpha)]$. In general, the endpoint of a range are not necessarily rational numbers, so we may not be able to represent these ranges exactly. Thus, what we want, in general, is to generate rational numbers which are ε -close to the desired endpoints $\underline{y}(\alpha)$ and $\bar{y}(\alpha)$, for some given rational number ε .

By definition of the range, we have

$$\underline{y}(\alpha) = \min_{x_1 \in \mathbf{x}_1(\alpha), \dots, x_n \in \mathbf{x}_n(\alpha)} f(x_1, \dots, x_n)$$

and

$$\bar{y}(\alpha) = \max_{x_1 \in \mathbf{x}_1(\alpha), \dots, x_n \in \mathbf{x}_n(\alpha)} f(x_1, \dots, x_n).$$

We are considering the same value α in all these formulas. Thus, we can safely skip the mentioning of this value – this will make all the notations clearer without introducing any confusion. Thus, we arrive at the following formulation of the problem.

Definition 1. *By the fuzzy computation problem, we mean the following problem:*

- given rational numbers $\underline{x}_1, \bar{x}_1, \dots, \underline{x}_n, \bar{x}_n$, $\varepsilon > 0$, and a quadratic function $f(x_1, \dots, x_n)$ with rational coefficients,
- compute rational numbers \underline{r} and \bar{r} which are ε -close to, correspondingly,

$$\underline{y} = \min_{\underline{x}_1 \leq x_1 \leq \bar{x}_1 \ \& \ \dots \ \& \ \underline{x}_n \leq x_n \leq \bar{x}_n} f(x_1, \dots, x_n) \text{ and}$$

$$\bar{y} = \max_{\underline{x}_1 \leq x_1 \leq \bar{x}_1 \ \& \ \dots \ \& \ \underline{x}_n \leq x_n \leq \bar{x}_n} f(x_1, \dots, x_n).$$

Proposition 1. *The fuzzy computation problem is NP-hard.*

Comment. The proof is, in effect, contained in [3], [14].

What we do in this paper. In this paper, we show that by having a more realistic formalization of fuzzy computing, we can make fuzzy computing feasible.

II. HOW TO MAKE FORMULAS OF FUZZY COMPUTING MORE REALISTIC

Main idea. The usual derivation of Zadeh’s extension principle for quantum computing consider *all* possible tuples (X_1, \dots, X_n) for which $f(X_1, \dots, X_n) = Y$. Similarly, in the formulas for the α -cut, we consider *all* possible tuples (x_1, \dots, x_n) for which $\mu_i(x_i) \geq \alpha$ for every i . In both cases, we took “all” literally: all means all, one exception makes a statement about all the tuples false.

From the mathematical viewpoint, this is a reasonable idea. But let us take into account that we are not proving mathematical theorems, we are trying to formalize common sense, we are trying to formalize expert reasoning. And in our usual reasoning, “all” does not literally mean mathematically

all, it usually means “almost all”, meaning everyone except a small fraction of the original population.

- When a patriotic journalist says that all the citizens of the country support it against its enemy – real or perceived – this all-statement is often followed by a confession that there are a few traitors who are willing to support the enemy.
- When we say that all pigeons can fly, we understand very well that there may be a wounded or deformed pigeon, but that most pigeons can fly.
- A classical AI example is a phrase “all birds fly” (see, e.g., [7]); this phrase has known exceptions, such as penguins, but the vast majority of the birds indeed can fly.

Let us see how the above definitions of fuzzy computing will change if we replace the mathematical meaning of “all” with a more commonsense meaning of this word.

Towards a new formalization of fuzzy computing. In line with what we have just discussed, instead of defining the desired upper endpoint \bar{y} as the exact maximum of all the values $f(x_1, \dots, x_n)$ when $x_i \in [\underline{x}_i, \bar{x}_i]$, it is reasonable to define \bar{y} as the maximum of “almost all” values.

If we fix the exact proportion $\delta > 0$ of values that we can ignore, that would mean that we are looking for a value \bar{y} for which

$$\frac{|\{x : x_1 \in \mathbf{x}_1 \ \& \ \dots \ \& \ x_n \in \mathbf{x}_n \ \& \ f(x_1, \dots, x_n) \leq \bar{y}\}|}{|\{x : x_1 \in \mathbf{x}_1 \ \& \ \dots \ \& \ x_n \in \mathbf{x}_n\}|} = 1 - \delta,$$

where $|S|$ denotes the multi-D volume of a set:

- width of an interval,
- area of a planar (2-D) set,
- volume of a 3-D set, etc.

When δ tends to 0, the corresponding value tends to the maximum of the function $f(x_1, \dots, x_n)$ on the box

$$\mathbf{x} \stackrel{\text{def}}{=} \mathbf{x}_1 \times \dots \times \mathbf{x}_n.$$

Thus, for small δ , the above-defined value is very close to this maximum.

Similarly, instead of defining the lower endpoint \underline{y} as the exact minimum of all the values $f(x)$ when $x \in \mathbf{x}$, it is reasonable to define \underline{y} as the value for which

$$\frac{|\{x : x \in \mathbf{x} \ \& \ f(x) \geq \underline{y}\}|}{|\{x : x \in \mathbf{x}\}|} = 1 - \delta.$$

When δ tends to 0, the corresponding value tends to the minimum of the function $f(x_1, \dots, x_n)$ on the box \mathbf{x} . Thus, for small δ , the above-defined value is very close to this minimum.

What value δ should we choose? Intuitively, since we are considering the *fuzzy* case, it makes no sense to fix one *exact* value δ , it is more appropriate to assume that this value is also given with some uncertainty – at least with interval uncertainty. In other words, it makes sense to assume that we know the interval $[\underline{\delta}, \bar{\delta}]$, with $\underline{\delta} < \bar{\delta}$, that contains the actual (unknown)

value δ . In this case, all we know about $1 - \delta$ is that it is somewhere between $1 - \bar{\delta}$ and $1 - \underline{\delta}$:

$$1 - \bar{\delta} \leq 1 - \delta \leq 1 - \underline{\delta}.$$

Thus, e.g., from the above equality for \bar{y} we get the double inequality:

$$1 - \bar{\delta} \leq \frac{|\{x : x \in \mathbf{x} \& f(x) \leq \bar{y}\}|}{|\{x : x \in \mathbf{x}\}|} \leq 1 - \underline{\delta}.$$

A similar inequality holds for y .

What does it mean to compute y and \bar{y} ? Since we relaxed the requirement on the endpoints y and \bar{y} – by allowing some probability that these values are not mathematically universal bounds – it makes sense to also relax the usual requirement on the algorithm: that it always computes the desired value. Specifically, from the practical viewpoint, it makes perfect sense to consider algorithms that provide an answer with a probability $1 - p_0$, for some small positive value $p_0 \ll 1$.

Indeed, even the computer hardware is not 100% reliable, once in a while computers break down. From this viewpoint, it is perfectly OK if the algorithm also sometimes does not produce the desired result – as long as the probability for this is much smaller than the probability of a hardware fault and thus, does not significantly increase the probability of an erroneous result.

Thus, we arrive at the following definitions.

III. DEFINITIONS OF THE MORE REALISTIC FUZZY COMPUTATIONS AND THE MAIN RESULT: THAT THE CORRESPONDING PROBLEM IS NOW FEASIBLE

Definition 2. Let $\varepsilon > 0$ be a rational number. We say that a function $f(x_1, \dots, x_n)$ is ε -feasible if there exists a feasible algorithm that, given rational values x_1, \dots, x_n , produces a rational number which is ε -close to $f(x_1, \dots, x_n)$.

Definition 3. Let $\varepsilon > 0$, $0 < \underline{\delta} < \bar{\delta}$, and $p_0 > 0$ be rational numbers. By realistic fuzzy computations, we mean the following problem:

- given rational numbers $\underline{x}_1, \bar{x}_1, \dots, \underline{x}_n, \bar{x}_n$, and an ε -feasible function $f(x_1, \dots, x_n)$ with rational coefficients,
- compute, with probability $\geq 1 - p_0$, rational numbers \underline{r} and \bar{r} which are ε -close to, correspondingly, values y and \bar{y} for which

$$1 - \bar{\delta} \leq \frac{|\{x : x \in \mathbf{x} \& f(x) \geq \underline{y}\}|}{|\{x : x \in \mathbf{x}\}|} \leq 1 - \underline{\delta} \text{ and}$$

$$1 - \bar{\delta} \leq \frac{|\{x : x \in \mathbf{x} \& f(x) \leq \bar{y}\}|}{|\{x : x \in \mathbf{x}\}|} \leq 1 - \underline{\delta}.$$

Proposition 2. For each tuple $(\varepsilon, \underline{\delta}, \bar{\delta}, p_0)$, there exists a feasible algorithm that solves the corresponding realistic fuzzy computations problem.

Proof. The desired algorithm uses the standard random number generator algorithm that generates numbers ξ uniformly distributed on the interval $[0, 1]$. For each interval $[\underline{x}_i, \bar{x}_i]$, by

using such a random number generator, we can compute the value $x_i = \underline{x}_i + \xi \cdot (\bar{x}_i - \underline{x}_i)$ which is uniformly distributed on the interval $[\underline{x}_i, \bar{x}_i]$. If we repeat this procedure n times, for n intervals $\mathbf{x}_i = [\underline{x}_i, \bar{x}_i]$, then we get a tuple (x_1, \dots, x_n) which is uniformly distributed on the box $\mathbf{x}_1 \times \dots \times \mathbf{x}_n$.

Now, we can formulate the resulting algorithm:

- first, we select an appropriate natural number N , and compute the values

$$\tilde{\delta} \stackrel{\text{def}}{=} \frac{\delta + \bar{\delta}}{2}, \quad \Delta \stackrel{\text{def}}{=} \frac{\bar{\delta} - \delta}{2}, \text{ and } v = \lfloor N \cdot \tilde{\delta} \rfloor;$$

- then, N times we use the above procedure for generating tuples uniformly distributed on the box $\mathbf{x}_1 \times \dots \times \mathbf{x}_n$, and get N tuples $x^{(1)}, \dots, x^{(N)}$;
- we apply the given feasible algorithm f to each of these tuples, generating N values $y^{(k)}$ which are ε -close to $f(x^{(k)})$:

$$|y^{(k)} - f(x^{(k)})| \leq \varepsilon;$$

- we sort these N values $y^{(k)}$ into an increasing sequence

$$y_{(1)} \leq y_{(2)} \leq \dots \leq y_{(N)};$$

- finally, we take $\underline{r} = y_{(v)}$ and $\bar{r} = y_{(N-v)}$.

Let us show that for an appropriate value N , this Monte-Carlo-type algorithm indeed solves the corresponding realistic fuzzy computation problem.

Indeed, each value $y^{(k)}$ is ε -close to the corresponding value $f(x^{(k)})$. Thus, if we sort the values $f(x^{(k)})$ into an increasing sequence $f_{(1)} \leq f_{(2)} \leq \dots \leq f_{(N)}$, then we get $|y_{(k)} - f_{(k)}| \leq \varepsilon$ for all k . In particular, we have

$$|y_{(v)} - f_{(v)}| \leq \varepsilon \text{ and } |y_{(N-v)} - f_{(N-v)}| \leq \varepsilon.$$

We will show that the desired inequalities hold for $\underline{y} = f_{(v)}$ and $\bar{y} = f_{(N-v)}$.

For each y , let us denote the probability that $f(x) \leq y$ under uniform distribution – i.e., the ratio

$$\frac{|\{x : x \in \mathbf{x} \& f(x) \leq y\}|}{|\{x : x \in \mathbf{x}\}|},$$

by p . Then, among N randomly selecting tuples x , the mean $E = E[V]$ of number V of tuples for which $f(x) \leq y$ is equal to $N \cdot p$. The number V is equal to the sum of N independent identically distributed variables η_1, \dots, η_N each of which is equal:

- to 1 if $f(x) \leq y$ and
- to 0 if $f(x) > y$.

For each such variable η , the variance is equal to

$$E[(\eta - E[\eta])^2] = E[(\eta - p)^2] =$$

$$p \cdot (1 - p)^2 + (1 - p) \cdot (0 - p)^2 = p \cdot (1 - p).$$

Thus, the variance of V is equal to $N \cdot p \cdot (1 - p)$, and thus, the standard deviation is equal to $\sigma[V] = \sqrt{N \cdot p \cdot (1 - p)}$.

For large N , due to the Central Limit Theorem (see, e.g., [13]), the distribution of V is close to Gaussian. Thus, for

an appropriate k_0 (whose value depend on p_0 [13]), with probability $\geq 1 - p_0$, the actual value of V is within the interval

$$[E[V] - k_0 \cdot \sigma[V], E[V] + k_0 \cdot \sigma[V]].$$

For example:

- for $p_0 = 0.05$, we can take $k_0 = 2$;
- for $p_0 = 0.001$, we can take $k_0 = 3$, and
- for $p_0 = 10^{-8}$, we can take $k_0 = 6$.

With the probability $\geq 1 - p_0$, the difference between the actual value V and the value $E[V] = p \cdot N$ does not exceed

$$k_0 \cdot \sqrt{N} \cdot \sqrt{p \cdot (1 - p)}.$$

The largest possible value of the product $p \cdot (1 - p)$ is attained when $p = 0.5$; in this case, $\sqrt{p \cdot (1 - p)} = 0.5$. Thus, we always have $|V - p \cdot N| \leq 0.5k_0 \cdot \sqrt{N}$. Hence,

$$\left| p - \frac{V}{N} \right| \leq \frac{0.5k_0}{\sqrt{N}}.$$

For $\underline{y} = f(v)$, we have $V = v$ and thus, the actual probability p that $f(x)$ is smaller than or equal to this value \underline{y} satisfies the inequality

$$\left| p - \frac{v}{N} \right| \leq \frac{0.5k_0}{\sqrt{N}}.$$

Here, by definition of v , we have $|v - \tilde{\delta} \cdot N| \leq 1$, hence

$$\left| \frac{v}{N} - \tilde{\delta} \right| \leq \frac{1}{N}, \text{ thus}$$

$$|p - \tilde{\delta}| \leq \left| p - \frac{v}{N} \right| + \left| \frac{v}{N} - \tilde{\delta} \right| \leq \frac{0.5k_0}{\sqrt{N}} + \frac{1}{N}.$$

We want to make sure that p is in the interval $[\delta, \bar{\delta}]$. This is equivalent to requiring that p differs from the center $\tilde{\delta}$ by no more than its half-width Δ , i.e., that $|p - \tilde{\delta}| \leq \Delta$. Thus, to make sure that our algorithm solves the corresponding problem, it is sufficient to find the value N for which

$$\frac{0.5k_0}{\sqrt{N}} + \frac{1}{N} \leq \Delta.$$

Such a value N is easy to find. For this N , the selected value \underline{y} satisfies the first of the two desired inequalities.

Similarly, we can prove that for this same N , the selected value $\bar{y} = f(N - v)$ satisfies the second of these inequalities. The proposition is proven.

Comment 1. The appearance of Monte-Carlo techniques in a paper about fuzzy computation may seem to contradict the fact that many papers on fuzzy computations emphasize that their results are faster to compute than more traditional Monte-Carlo-based techniques. However, there is no contradiction; indeed:

- the computation time of usual fuzzy computation algorithms grows with n , while
- the number of iterations N needs for a Monte-Carlo algorithm does not depend on n at all – and the above proof is an example of this fact.

As a result, for large n , when the computation time of the usual fuzzy computation algorithms grows very large, Monte-Carlo-type methods become much more efficient. But, on the other hand, when the number of inputs n is reasonably small, the usual methods are much faster – which is exactly what many papers have claimed.

In this paper, our main objective is to find a feasible algorithm, i.e., an algorithm that works reasonably well even for large n . This is exactly why we produced the above algorithm. By no means is this algorithm optimal – especially for small n , when clearly the usual methods are faster.

Comment 2. In this paper, we provide a feasible algorithm for a realistic formulation of maximization and minimization problems. However, similar ideas can be applied to come up with feasible algorithms for solving more complex problems, such as minimax or maximin, where we need to compute

$$\min_{x \in X} \max_{y \in Y} f(x, y) \text{ or } \max_{x \in X} \min_{y \in Y} f(x, y);$$

this is important, e.g., in game theory; see, e.g., [5]. Such problems are also, in general, NP-hard, but if we replace maximum and minimum over all possible value to maximum and minimum over almost all values, then we will be able to produce similar feasible algorithms. Same idea can be applied to more general problems, when we have more than two combinations of minimum and maximum.

These problems appear, e.g., if we start with a general first-order formula – i.e., a formula obtained from equalities and inequalities by applying logical connectives “and”, “or”, and “not” and quantifiers $\exists x$ and $\forall x$ running over appropriate intervals. Indeed:

- each strict inequality $f > g$ can be approximated, with any possible accuracy $\delta > 0$, as a non-strict inequality

$$f \geq g + \delta;$$

- each equality $f = g$ can be represented as two inequalities $f \geq g$ and $g \geq f$;
- each inequality $f \geq g$ can be represented as $h \geq 0$, for

$$h \stackrel{\text{def}}{=} f - g;$$

- the formula $f \geq 0 \& g \geq 0$ can be represented as

$$\min(f, g) \geq 0;$$

- the formula $f \geq 0 \vee g \geq 0$ can be represented as

$$\max(f, g) \geq 0;$$

- the formula $\exists x (f \geq 0)$ is equivalent to $\max_x f \geq 0$; and
- the formula $\forall x (f \geq 0)$ is equivalent to $\min_x f \geq 0$.

REFERENCES

- [1] R. Belohlavek, J. W. Dauben, and G. J. Klir, *Fuzzy Logic and Mathematics: A Historical Perspective*, Oxford University Press, New York, 2017.
- [2] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

- [3] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Kluwer, Dordrecht, 1998.
- [4] J. M. Mendel, *Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions*, Springer, Cham, Switzerland, 2017.
- [5] R. B. Myerson, *Game Theory: Analysis of Conflict*, Harvard University Press, Harvard, Massachusetts, 1997.
- [6] H. T. Nguyen, C. Walker, and E. A. Walker, *A First Course in Fuzzy Logic*, Chapman and Hall/CRC, Boca Raton, Florida, 2019.
- [7] N. Nilsson, *Artificial Intelligence: A New Synthesis*, Morgan Kaufman, San Francisco, California, 2003.
- [8] V. Novák, I. Perfilieva, and J. Močkoř, *Mathematical Principles of Fuzzy Logic*, Kluwer, Boston, Dordrecht, 1999.
- [9] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, San Diego, 1994.
- [10] E. D. Rodriguez Velasquez, C. M. Chang Albitres, and V. Kreinovich, "Fuzzy ideas explain a complex heuristic algorithm for gauging pavement conditions", *Mathematical Structures and Modeling*, 2018, Vol. 47, pp. 82–90.
- [11] E. D. Rodriguez Velasquez, C. M. Chang Albitres, and V. Kreinovich, "Measurement-type 'calibration' of expert estimates improves their accuracy and their usability: pavement engineering case", *Proceedings of the IEEE Symposium on Computational Intelligence for Engineering Solutions CIES'2018*, Bengaluru, India, November 18–21, 2018.
- [12] E. D. Rodriguez Velasquez, C. M. Chang Albitres, Thach Ngoc Nguyen, O. Kosheleva, and V. Kreinovich, "How to take expert uncertainty into account: economic approach illustrated by pavement engineering applications", In: V. Kreinovich and S. Sriboonchitta (eds.), *Structural Changes and Their Econometric Modeling*, Springer Verlag, Cham, Switzerland, 2019.
- [13] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman and Hall/CRC, Boca Raton, Florida, 2011.
- [14] S. A. Vavasis, *Nonlinear Optimization: Complexity Issues*, Oxford University Press, New York, 1991.
- [15] L. A. Zadeh, "Fuzzy sets", *Information and Control*, 1965, Vol. 8, pp. 338–353.