6-1-2022

Efficient Algorithms for Data Processing under Type-3 (and Higher) Fuzzy Uncertainty

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Olga Kosheleva
The University of Texas at El Paso, olgak@utep.edu

Patricia Melin
Tijuana Institute of Technology, pmelin@tectijuana.mx

Oscar Castillo
Tijuana Institute of Technology, ocastillo@tectijuana.mx

Comments:

Technical Report: UTEP-CS-22-66

Efficient Algorithms for Data Processing under Type-3 (and Higher) Fuzzy Uncertainty

Vladik Kreinovich[1], Olga Kosheleva[1],
Patricia Melin[2], and Oscar Castillo[2]
[1]University of Texas at El Paso, 500 W. University,
El Paso, Texas 79968, USA, olgak@utep.edu, vladik@utep.edu
[2]Tijuana Institute of Technology, Tomas Aquino,
Baja California, Mexico,
pmelin@tectijuana.mx, ocastillo@tectijuana.mx

Abstract

It is known that to more adequately describe expert knowledge, it is necessary to go from the traditional (type-1) fuzzy techniques to higher order ones: type-2, probably type-3 and even higher. Until recently, only type-1 and type-2 fuzzy sets were used in practical applications. However, lately, it turned out that type-3 fuzzy sets are also useful in some applications. Because of this practical importance, it is necessary to design efficient algorithms for data processing under such type-3 (and higher order) fuzzy uncertainty. In this paper, we show how we can combine known efficient algorithms for processing type-1 and type-2 uncertainty to come up with a new algorithm for the type-3 case.

1 Outline

Usual data processing algorithms treat data points as if they were exact. In practice, data comes with uncertainty. When data comes from experts who describe their knowledge by using imprecise ("fuzzy") words from natural language, a natural way to describe the corresponding uncertainty is to use fuzzy techniques. To get a more accurate representation of expert uncertainty, it is necessary to use higher-order fuzzy techniques, i.e., go from the usual $[0, 1]$-based type-1 techniques to type-2, type-3, and maybe even higher types. In this paper, we describe efficient algorithms for data processing under such higher-order fuzzy uncertainty.

The structure of this paper is as follows. In Section 2, we recall the need for data processing. In Section 3, we recall the need for fuzzy techniques and for higher-order fuzzy techniques. In Sections 4, 5, and 6, we recall how data can be processed under type-1, interval type-2, and general type-2 fuzzy uncertainty.

Finally, in Section 7, we use these known results to come up with new efficient algorithms for data processing under type-3 and higher order fuzzy uncertainty.

2 Why Data Processing

One of the main objectives of science is to describe the current state of the world – and to predict its future state. One of the main objectives of engineering is to design new buildings, gadgets, and/or new algorithms to make this future better. To describe the state of the world – and to describe the engineered objects – we need to list the numerical values of the quantities that characterize different natural and artificial objects.

Some quantities we can simply measure: we can directly measure the temperature outside, we can directly measure the distance between the two nearby buildings, etc. However, many quantities we cannot measure directly: e.g., we cannot directly measure the distance to a faraway star or the amount of oil in a given oilfield. And it is definitely not possible to directly measure the future state – e.g., future temperature. To estimate such a difficult-to-measure quantity y, a natural idea is to find easier-to-measure-or-estimate quantities $x_1, \ldots, x_n$ that are related to the desired quantity y by a known dependence $y = f(x_1, \ldots, x_n)$. Then, we can measure or estimate the quantities x_i, and use the results $\widetilde{y}$ of measurement or estimation to estimate y as $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

Computing this estimate, i.e., applying the algorithm $f(x_1, \ldots, x_n)$ to the results of measurements and/or expert estimations is what is usually called *data processing.*

3 Need for Fuzzy Uncertainty and Need for Higher-Order Fuzzy Uncertainty

Need for fuzzy uncertainty. Often, estimates for x_i come from experts, and experts rarely provide exact values. Expert knowledge is usually formulated by using imprecise ("fuzzy") words from natural language. An experienced driver explaining his/her driving strategy will not say that in a certain situation, you need to show down by exactly 5.0 km/h, he/she will probably say "slow down a little bit", or "slow down by about 5 km/h".

We want to use this imprecise knowledge in computer-related data processing. The challenge is that computers were designed to process numbers, not words from natural language. So, we need to transform expert statements into computer-understandable numerical form. For this purpose, Lotfi Zadeh invented fuzzy techniques (see, e.g., [1, 7, 12, 14, 15, 16]), where each imprecise term like "small" is described by assigning, to each possible value x of the corresponding quantity, the degree $m(x)$ – from the interval $[0, 1]$ – to which, according to the expert, this value is small. The resulting function $m(x)$ is known as the *membership function* or, alternatively, as the *fuzzy set.* This original idea is also called *type-1 fuzzy techniques.*

Fuzzy numbers. For most terms, the membership function first (non-strictly) increases then (non-strictly) decreases. Such membership functions are known as *fuzzy numbers.*

"And"- and "or"-operations (t-norms and t-conorms). Expert rules often involve logical connectives like "and" and "or". For example, a rule can say that if a car in front of you is close *and* it slows down a little bit, then you should break a little bit. Strictly speaking, in this case, we need to find out, for each pair consisting of a distance value and a change-in-velocity value, the degree to which, for this pair, the condition "a car in front of you is close *and* it slows down a little bit" is satisfied. In this case, we may be able to do it, but, e.g., in medicine, we have rules with 5 or 6 different conditions. Even if we try only 10 values for each of the 5-6 variables, this still means asking 10^5 to 10^6 questions to an expert – this is not feasible. In such situations, to estimate the degree of confidence in a composite statement $A\,\&\,B$ or $A \vee B$, the only information we have is the expert's degrees of confidence a and b in the original statements A and B.

The algorithm $f_\&(a,b)$ that estimates the degree of confidence in $A\,\&\,B$ based on this information is known as an *"and"-operation* or, for historical reason, a *t-norm.* Similarly, the algorithm $f_\vee(a,b)$ that estimates the degree of confidence in $A \vee B$ based on this information is known as an *"or"-operation* or, for historical reason, a *t-conorm.* The simplest – and frequently used – "and"- and "or"-operations are $f_\&(a,b) = \min(a,b)$ and $f_\vee(a,b) = \max(a,b)$.

Need for type-2 fuzzy technique. The challenge with type-1 fuzzy technique is that similarly to the fact that an expert cannot name the exact value of the quantity, the same expert cannot produce the exact degree $m(x)$. At best, the expert can provide an interval of possible values of this degree – e.g., $[0.6, 0.7]$ – or even a fuzzy statement like "the degree is close to 0.6". So, a natural idea is to allow the degree $m(x)$ to be an interval – which leads to *interval-valued fuzzy sets* – or even a fuzzy number corresponding to a statement like "the degree is close to 0.6" – this leads to so-called *type-2 fuzzy sets.*

In general, an interval $[\underline{x}, \overline{x}]$ can be viewed as a fuzzy set – the degree of confidence is 1 for all the values inside this interval and 0 for all the values outside this interval. Thus, interval-values fuzzy sets are particular cases of type-2 fuzzy sets.

In the interval-valued case, the expert-generated degree of confidence is an interval $\mathbf{m}(x) = [\underline{m}(x), \overline{m}(x)]$. In the general type-2 case, for each number t from the interval $[0,1]$, the expert provides a degree to which this number t is a degree of confidence that x has the desired property (like "small"). We will denote this degree by $m(x,t)$.

Need for type-3 and higher-order fuzzy techniques. Similarly to the fact that an expert cannot describe his/her degree of confidence – that x is small – by a single number, the same expert cannot describe his/her degree of confidence that t is a degree of confidence that x is small by a single number. At best, the expert can provide either an interval $[\underline{m}(x,t), \overline{m}(x,t)]$ or a fuzzy number that describes this degree of confidence. The fuzzy case is known as *type-3 fuzzy technique*, and the interval-valued case is known as *interval type-3*.

In the general type-3 case, for each value s from the interval $[0,1]$, we provide a degree – denoted by $m(x,t,s)$ – that s is degree of confidence in the statement "t is a degree of confidence that x has the desired property".

Is this worth considering? At first glance, the difference between type-2 and type-3 is so subtle and complicated that one can doubt whether it is necessary to use type-3 in practical applications. Actually, people doubted that type-2 would be practically useful – and it turned out that it is often useful; see, e.g., [12]. Similarly, it turned out that type-3 techniques are also useful in many practical cases; see, e.g., [2, 3, 4, 5] and references therein.

What about higher order types? Clearly, an expert cannot provide the exact degree $m(x,t,s)$, so a natural idea is to allow an expert to provide interval-valued of fuzzy degrees – which leads to type-4, where for each real number r from the interval $[0,1]$, we ask the expert to describe his/her degree of confidence $m(x,t,s,r)$ that r is a proper value of $m(x,t,s)$.

The expert cannot describe the precise value of $m(x,t,s,r)$, so this value can also be fuzzy – we get type-5, etc.

Need for data processing under such uncertainty. Since type-1, type-2, and type-3 fuzzy techniques are practically useful, it is desirable to develop efficient algorithms for data processing under such uncertainty. Efficient algorithms for type-1 and type-2 are known – we describe them in the following sections. Efficient algorithms for type-3 case are described in the last section of this paper.

We do not know yet whether type-4, type-5, etc., will be practically useful, but the fact that type-2 and type-3 turned out to be useful makes us think that it is quite probable that higher-order fuzzy sets will be useful. So it makes sense to think of efficient algorithms for these cases too, and this is what we will do in the same last section.

4 Data Processing under Type-1 Fuzzy Uncertainty: Reminder

Formulation of the problem: reminder.

- We know that the quantity-of-interest y is a function $y = f(x_1,\ldots,x_n)$ of several auxiliary quantities $x_1,\ldots,x_n$.
- We also know, for each i, the membership function $m_i(x_i)$ that describes, for each real number x_i, the degree to which this number is a possible value of the i-th input.

Based on this information, we want to describe, for each real number y, the degree $m(y)$ to which this number is a possible value of the quantity of interest.

Zadeh's extension principle: derivation and the resulting formula. A value y is possible if $y = f(x_1, \ldots, x_n)$ for some possible values x_i. We know the degree $m_i(x_i)$ to which each value x_i is possible. We can therefore use the min "and"-operation to describe, for each tuple $(x_1, \ldots, x_n)$ for which $y = f(x_1, \ldots, x_n)$, the degree to which all its values are possible – i.e.. x_1 is possible *and* x_2 is possible, etc. – as $\min(m_1(x_1), \ldots, m_n(x_n))$.

The value y if possible if either the first tuple $(x_1, \ldots, x_n)$ for which $y = f(x_1, \ldots, x_n)$ is possible, *or* the second such tuple is possible, etc. We can therefore us the max "or"-operation to estimate the degree to which y is possible as

$$m(y) = \sup\{\min(m_1(x_1), \ldots, m_n(x_n)) : y = f(x_1, \ldots, x_n)\}. \tag{1}$$

This formula was first described by Zadeh himself and is therefore known as *Zadeh's extension principle.*

How to actually compute this formula: analysis of the problem. Straightforward computation of the formula (1) requires solving a complex constraint optimization problem – which is, in general, time-consuming. It is known, however, that there are more efficient ways to compute $m(y)$. These ways are related to the notion of *α-cuts* of a fuzzy sets, which are defined, for each $\alpha \in (0, 1]$, as $\{x : m(x) \geq \alpha\}$. For fuzzy numbers, each α-cut is an interval; we will denote it by $\mathbf{x}(\alpha) = [\underline{m}(\alpha), \overline{m}(\alpha)]$.

For $\alpha = 0$, we can use a slightly different formulation of the α-cut: it the closure $\mathbf{x}(0) = \overline{\{x : m(x) > 0\}}$ of the set $\{x : m(x) > 0\}$. In the following text, for simplicity, we will only list the simpler formula which is valid for $\alpha > 0$, but, of course, for $\alpha = 0$, we have to use the more complex formula.

Once we know all the α-cuts, we can reconstruct the membership function as $m(x) = \sup\{\alpha : x \in \mathbf{x}(\alpha)\}$. In particular, if we know α-cuts for $\alpha = 0, 0.1, 0.2, \ldots, 1.0$, then we can reconstruct $m(x)$ with accuracy 0.1 – which is usually sufficient, since experts rarely produce their degree of confidence with higher accuracy. So, to find $m(y)$, it is sufficient to find the α-cuts $\mathbf{y}(\alpha)$ for the corresponding 11 values α.

Because of the possibility to easily move from the usual representation of the membership function $m(x)$ and its α-cut representation, sometime the membership function is stored by listing the corresponding α-cuts.

To find the α-cuts corresponding to the desired quantity y, we can take into account that the value $m(y)$ as described by the formula (1) is larger than or equal to α if and only if for one of the tuples $(x_1, \ldots, x_n)$ for which $y = f(x_1, \ldots, x_n)$, we have $\min(m_1(x_1), \ldots, m_n(x_n)) \geq \alpha$. This inequality, in its turn, is equivalent to requiring that $m_i(x_i) \geq \alpha$ for all i. Thus, the α-cut for y is equal to the range of the function $y = f(x_1, \ldots, x_n)$ when each x_i is in the corresponding α-cut:

$$\mathbf{y}(\alpha) = f(\mathbf{x}_1(\alpha), \ldots, \mathbf{x}_n(\alpha)), \tag{2}$$

where for each sets $X_1, \ldots, X_n$, the range $f(X_1, \ldots, X_n)$ is defined as

$$f(X_1, \ldots, X_n) \stackrel{\text{def}}{=} \{f(x_1, \ldots, x_n) : x_1 \in X_1, \ldots, x_n \in X_n\}. \tag{3}$$

The problem of computing the range of a function when each input is in a known interval is known as the problem of *interval computations*; there are efficient general algorithms for estimating this range, see, e.g., [6, 10, 11, 13]

Comment. In some important cases, interval computation is easy, no general complex algorithms are needed. For example, if the function $f(x_1,\ldots,x_n)$ is (non-strictly) increasing in each of its variables, then the smallest value of this function on intervals $X_i=[\underline{x}_i,\overline{x}_i]$ is attained when each input x_i is the smallest, i.e., when $x_i=\underline{x}_i$ for all i. Similarly, the largest value of this function on intervals $X_i=[\underline{x}_i,\overline{x}_i]$ is attained when each input x_i is the largest, i.e., when $x_i=\overline{x}_i$ for all i. Thus,

$$f([\underline{x}_1,\overline{x}_1],\ldots,[\underline{x}_n,\overline{x}_n])=[f(\underline{x}_1,\ldots,\underline{x}_n),f(\overline{x}_1,\ldots,\overline{x}_n)].$$

Resulting algorithm.

- First, if the information about the inputs x_i is stored in the form of the usual membership functions $m_i(x_i)$, we compute, for each i and for each value $\alpha\in\{0,0.1,\ldots,1.0\}$, the corresponding α-cut

  $$\mathbf{x}_i(\alpha)=\{x_i : m_i(x_i)\ge\alpha\}.$$

  (Recall that for $\alpha=0$, we will have to use a slightly more complex formula.)

- Then, for each value α from the above list, we use an interval computation algorithm to compute the range $\mathbf{y}(\alpha)=f(\mathbf{x}_1(\alpha),\ldots,\mathbf{x}_n(\alpha))$. These ranges form the α-cut representation of the desired membership function $m(y)$.

- Finally, if we want to represent this membership function in the usual form, we compute $m(y)=\max\{y : y\in\mathbf{y}(\alpha)\}$.

How many computation steps do we need. These computations need to be repeated for all α. So, if we use 11 values $\alpha=0,0.1,\ldots,1.0$, then, to find the result of data processing under type-1 fuzzy uncertainty, we need to apply an interval computations algorithm 11 times.

5 Data Processing under Interval-Valued Fuzzy Uncertainty: Reminder

Formulation of the problem. In the interval-valued case, the relation between $m(y)$ and $m_i(x_i)$ is described by the same formula (1); the main difference is that now, values $m(y)$ and $m_i(x_i)$ are not numbers but intervals.

The corresponding efficient algorithms are described in [8, 9].

Interval case: analysis of the problem. In the interval case, each value $m_i(x_i)$ is an interval $[\underline{m}_i(x_i),\overline{m}_i(x_i)]$. The right-hand side of the formula (1) is a non-strictly increasing function of all the values $m_i(x_i)$. Thus, the desired range is equal to $[\underline{m}(y), \overline{m}(y)]$, where

$$\underline{m}(y) = \sup\{\min(\underline{m}_1(x_1), \ldots, \underline{m}_n(x_n)) : y = f(x_1, \ldots, x_n)\} \text{ and}$$

$$\overline{m}(y) = \sup\{\min(\overline{m}_1(x_1), \ldots, \overline{m}_n(x_n)) : y = f(x_1, \ldots, x_n)\}.$$

These are exactly formulas (1) for membership functions $\underline{m}_i(x_i)$ and $\overline{m}_i(x_i)$. So, to compute each of the two bounds $\underline{m}(y)$ and $\overline{m}(y)$, we can use the efficient α-cut-based algorithm.

Interval case: resulting algorithm. We are given interval-valued membership functions $[\underline{m}_i(x_i), \overline{m}_i(x_i)]$.

- Based on each of these membership functions, for each i and for each value α from the given list, we compute the orrepsonding α-cuts as:

  $$\underline{\mathbf{x}}_i(\alpha) = \{x_i : \underline{m}_i(x_i) \geq \alpha\} \text{ and } \overline{\mathbf{x}}_i(\alpha) = \{x_i : \overline{m}_i(x_i) \geq \alpha\}.$$

- We compute the α-cuts $\underline{\mathbf{y}}(\alpha)$ and $\overline{\mathbf{y}}(\alpha)$ for the endpoints $\underline{m}(y)$ and $\overline{m}(y)$ of the interval-valued membership function $[\underline{m}(y), \overline{m}(y)]$ as follows:

  $$\underline{\mathbf{y}}(\alpha) = f(\underline{\mathbf{x}}_1(\alpha), \ldots, \underline{\mathbf{x}}_n(\alpha)) \text{ and } \overline{\mathbf{y}}(\alpha) = f(\overline{\mathbf{x}}_1(\alpha), \ldots, \overline{\mathbf{x}}_n(\alpha)).$$

- Finally, the compute the endpoints $\underline{m}(y)$ and $\overline{m}(y)$ of the desired interval-valued membership function $[\underline{m}(y), \overline{m}(y)]$ as

  $$\underline{m}(y) = \max\{y : y \in \underline{\mathbf{y}}(\alpha)\} \text{ and } \overline{m}(y) = \max\{y : y \in \overline{\mathbf{y}}(\alpha)\}.$$

How many computation steps do we need. These computations need to be repeated for all α. So, if we use 11 values $\alpha = 0, 0.1, \ldots, 1.0$, then, to find the result of data processing under type-2 fuzzy uncertainty, we need to apply an interval computations algorithm $2 \cdot 11 = 22$ times.

6 Data Processing under General Type-2 Fuzzy Uncertainty: Reminder

Formulation of the problem. In the general type-2 case, the relation between $m(y)$ and $m_i(x_i)$ is described by the same formula (1); the main difference is that now, values $m(y)$ and $m_i(x_i)$ are not numbers but fuzzy sets.

The corresponding efficient algorithms are described in [8, 9].

General type-2 case: analysis of the problem. In the general type-2 case, $m(y)$ and $m_i(x_i)$ are fuzzy numbers. In this case, we can use the general type-1 result that the processing of fuzzy numbers is equivalent to computing the ranges of the processing function on different α-cuts. In this case, the data processing is described by the formula (1).

To distinguish α-cuts of the original membership functions for x_i and y and the α-cuts of each fuzzy number $m(y)$ and $m_i(x_i)$, we will use the letter β for the new alpha-cuts. Thus, we get the following for each β:

$$\mathbf{m}(y)(\beta) = \sup\{\min(\mathbf{m}_1(x_1)(\beta), \ldots, \mathbf{m}_n(x_n)(\beta)) : y = f(x_1, \ldots, x_n)\},$$

where

$$\mathbf{m}(y)(\beta) \stackrel{\text{def}}{=} \{t : m(y,t) \geq \beta\} \text{ and } \mathbf{m}_i(x_i)(\beta) \stackrel{\text{def}}{=} \{t : m_i(x_i,t) \geq \beta\}.$$

For fuzzy numbers, β-cuts are intervals, and the corresponding relation (1) is increasing. Thus, the above formula means that to get the lower endpoint $\underline{m}(y)(\beta)$ of a y's β-cut, we need to use only lower endpoints for β-cuts for x_i, and similarly for the upper endpoints:

$$\underline{m}(y)(\beta) = \sup\{\min(\underline{m}_1(x_1)(\beta), \ldots, \underline{m}_n(x_n)(\beta)) : y = f(x_1, \ldots, x_n)\} \text{ and}$$

$$\overline{m}(y)(\beta) = \sup\{\min(\overline{m}_1(x_1)(\beta), \ldots, \overline{m}_n(x_n)(\beta)) : y = f(x_1, \ldots, x_n)\}.$$

Each of these formulas is, in effect, Zadeh's extension principle for the corresponding membership functions. Thus, there formulas can be reformulated in terms of α-cuts of the corresponding membership functions:

$$\underline{\mathbf{y}}(\alpha,\beta) = f(\underline{\mathbf{x}}_1(\alpha,\beta), \ldots, \underline{\mathbf{x}}_n(\alpha,\beta)) \text{ and}$$

$$\overline{\mathbf{y}}(\alpha,\beta) = f(\overline{\mathbf{x}}_1(\alpha,\beta), \ldots, \overline{\mathbf{x}}_n(\alpha,\beta)),$$

where

$$\underline{\mathbf{y}}(\alpha,\beta) \stackrel{\text{def}}{=} \{y : \underline{m}(y)(\beta) \geq \alpha\}, \quad \underline{\mathbf{x}}_i(\alpha,\beta) \stackrel{\text{def}}{=} \{x_i : \underline{m}_i(x_i)(\beta) \geq \alpha\},$$

$$\overline{\mathbf{y}}(\alpha,\beta) \stackrel{\text{def}}{=} \{y : \overline{m}(y)(\beta) \geq \alpha\}, \quad \overline{\mathbf{x}}_i(\alpha,\beta) \stackrel{\text{def}}{=} \{x_i : \overline{m}_i(x_i)(\beta) \geq \alpha\}.$$

Hence, we arrive at the following algorithm:

General type-2 case: resulting algorithm. We start with type-2 membership functions $m_i(x_i,t)$.

- First, for each i and for each value β from the given list, we compute the β-cuts
$$[\underline{m}_i(x_i)(\beta), \overline{m}_i(x_i)(\beta)] \stackrel{\text{def}}{=} \{t : m_i(x_i,t) \geq \beta\}.$$
- Then, for each i and for each pair of values (α,β) from the given list, we compute the α-cuts
$$\underline{\mathbf{x}}_i(\alpha,\beta) \stackrel{\text{def}}{=} \{x_i : \underline{m}_i(x_i)(\beta) \geq \alpha\} \text{ and } \overline{\mathbf{x}}_i(\alpha,\beta) \stackrel{\text{def}}{=} \{x_i : \overline{m}_i(x_i)(\beta) \geq \alpha\}.$$
- For each α and β, we then use an interval computation algorithm to compute:
$$\underline{\mathbf{y}}(\alpha,\beta) = f(\underline{\mathbf{x}}_1(\alpha,\beta), \ldots, \underline{\mathbf{x}}_n(\alpha,\beta)) \text{ and}$$
$$\overline{\mathbf{y}}(\alpha,\beta) = f(\overline{\mathbf{x}}_1(\alpha,\beta), \ldots, \overline{\mathbf{x}}_n(\alpha,\beta)).$$

- Based on these intervals, for each β, we compute
$$\underline{m}(y)(\beta) = \sup\{\alpha : y \in \underline{\mathbf{y}}(\alpha,\beta)\} \text{ and } \overline{m}(y)(\beta) = \sup\{\alpha : y \in \overline{\mathbf{y}}(\alpha,\beta)\}.$$
- Finally, we compute the desired membership function
$$m(y,t) = \max\{\beta : t \in [\underline{m}(y)(\beta), \overline{m}(y)(\beta)]\}.$$

How many computation steps do we need. These computations need to be repeated for all α and β. So, if for each of these two parameters, we use 11 values $\alpha, \beta = 0, 0.1, \ldots, 1.0$, then, to find the result of data processing under type-2 fuzzy uncertainty, we need to apply an interval computations algorithm $2 \cdot 11^2 = 242$ times.

7 Data Processing under Type-3 (and Higher Order) Fuzzy Uncertainty: A New Algorithm

Formulation of the problem. Let us show the above type-2 algorithms can be used to come with an efficient algorithm for the type-3 case.

Type-3 case: analysis of the problem. In the type-3 case, each value $m(y)$ and $m_i(x_i)$ is a type-2 fuzzy set. Thus, we have the relation (1) between these type-2 fuzzy sets. So, based on the algorithm presented in the previous section, for each pair of values β and γ from the interval $[0,1]$, we have:

$$\underline{\mathbf{m}}(y)(\beta,\gamma) =$$
$$\sup\{\min(\underline{\mathbf{m}}_1(x_1)(\beta,\gamma), \ldots, \underline{\mathbf{m}}_n(x_n)(\beta,\gamma)) : y = f(x_1,\ldots,x_n)\} \tag{4}$$

and

$$\overline{\mathbf{m}}(y)(\beta,\gamma) =$$
$$\sup\{\min(\overline{\mathbf{m}}_1(x_1)(\beta,\gamma), \ldots, \overline{\mathbf{m}}_n(x_n)(\beta,\gamma)) : y = f(x_1,\ldots,x_n)\}, \tag{5}$$

where

$$\underline{\mathbf{m}}(y)(\beta,\gamma) = [\underline{m}^-(y)(\beta,\gamma), \underline{m}^+(y)(\beta,\gamma)] \stackrel{\text{def}}{=} \{t : \underline{m}(y,t)(\gamma) \ge \beta\},$$

$$\underline{\mathbf{m}}_i(x_i)(\beta,\gamma) = [\underline{m}_i^-(x_i)(\beta,\gamma), \underline{m}_i^+(x_i)(\beta,\gamma)] \stackrel{\text{def}}{=} \{t : \underline{m}_i(x_i,t)(\gamma) \ge \beta\},$$

$$\overline{\mathbf{m}}(y)(\beta,\gamma) = [\overline{m}^-(y)(\beta,\gamma), \overline{m}^+(y)(\beta,\gamma)] \stackrel{\text{def}}{=} \{t : \overline{m}(y,t)(\gamma) \ge \beta\},$$

$$\overline{\mathbf{m}}_i(x_i)(\beta,\gamma) = [\overline{m}_i^-(x_i)(\beta,\gamma), \overline{m}_i^+(x_i)(\beta,\gamma)] \stackrel{\text{def}}{=} \{t : \overline{m}_i(x_i,t)(\gamma) \ge \beta\},$$

and

$$[\underline{m}(y,t)(\gamma), \overline{m}(y,t)(\gamma)] \stackrel{\text{def}}{=} \{s : m(y,t,s) \ge \gamma\},$$

$$[\underline{m}_i(x_i,t)(\gamma), \overline{m}_i(x_i,t)(\gamma)] \stackrel{\text{def}}{=} \{s : m_i(x_i,t,s) \ge \gamma\}.$$

The corresponding transformation (1) is non-strictly increasing, thus the formulas (4) and (5) lead to similar relations between endpoints of the corresponding intervals:

$$\underline{m}^-(y)(\beta,\gamma) = \sup\{\min(\underline{m}_1^-(x_1)(\beta,\gamma),\ldots,\underline{m}_n^-(x_n)(\beta,\gamma)) : y = f(x_1,\ldots,x_n)\}, \tag{6}$$

$$\underline{m}^+(y)(\beta,\gamma) = \sup\{\min(\underline{m}_1^+(x_1)(\beta,\gamma),\ldots,\underline{m}_n^+(x_n)(\beta,\gamma)) : y = f(x_1,\ldots,x_n)\}, \tag{7}$$

$$\overline{m}^-(y)(\beta,\gamma) = \sup\{\min(\overline{m}_1^-(x_1)(\beta,\gamma),\ldots,\overline{m}_n^-(x_n)(\beta,\gamma)) : y = f(x_1,\ldots,x_n)\}, \tag{8}$$

$$\overline{m}^+(y)(\beta,\gamma) = \sup\{\min(\overline{m}_1^+(x_1)(\beta,\gamma),\ldots,\overline{m}_n^+(x_n)(\beta,\gamma)) : y = f(x_1,\ldots,x_n)\}. \tag{9}$$

Each of the formulas (6)–(9) is, in effect, Zadeh's extension principle for the corresponding membership functions. Thus, there formulas can be reformulated in terms of α-cuts of the corresponding membership functions:

$$\underline{\mathbf{y}}^-(\alpha,\beta,\gamma) = f(\underline{\mathbf{x}}_1^-(\alpha,\beta,\gamma),\ldots,\underline{\mathbf{x}}_n^-(\alpha,\beta,\gamma)),$$

$$\underline{\mathbf{y}}^+(\alpha,\beta,\gamma) = f(\underline{\mathbf{x}}_1^+(\alpha,\beta,\gamma),\ldots,\underline{\mathbf{x}}_n^+(\alpha,\beta,\gamma)),$$

$$\overline{\mathbf{y}}^-(\alpha,\beta,\gamma) = f(\overline{\mathbf{x}}_1^-(\alpha,\beta,\gamma),\ldots,\overline{\mathbf{x}}_n^-(\alpha,\beta,\gamma)),$$

$$\overline{\mathbf{y}}^+(\alpha,\beta,\gamma) = f(\overline{\mathbf{x}}_1^+(\alpha,\beta,\gamma),\ldots,\overline{\mathbf{x}}_n^+(\alpha,\beta,\gamma)),$$

where

$$\underline{\mathbf{y}}^-(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{y : \underline{m}^-(y)(\beta,\gamma) \ge \alpha\},\quad \underline{\mathbf{x}}_i^-(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{x_i : \underline{m}_i^-(x_i)(\beta,\gamma) \ge \alpha\},$$

$$\underline{\mathbf{y}}^+(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{y : \underline{m}^+(y)(\beta,\gamma) \ge \alpha\},\quad \underline{\mathbf{x}}_i^+(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{x_i : \underline{m}_i^+(x_i)(\beta,\gamma) \ge \alpha\},$$

$$\overline{\mathbf{y}}^-(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{y : \overline{m}^-(y)(\beta,\gamma) \ge \alpha\},\quad \overline{\mathbf{x}}_i^-(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{x_i : \overline{m}_i^-(x_i)(\beta,\gamma) \ge \alpha\},$$

$$\overline{\mathbf{y}}^+(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{y : \overline{m}^+(y)(\beta,\gamma) \ge \alpha\},\quad \overline{\mathbf{x}}_i^+(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{x_i : \overline{m}_i^+(x_i)(\beta,\gamma) \ge \alpha\}.$$

Hence, we arrive at the following algorithm.

Type-3 case: resulting algorithm. We start with type-3 membership functions $m_i(x_i,t,s)$.

- First, for every i and for all γ from the selected list of values, we compute:

$$[\underline{m}_i(x_i,t)(\gamma),\overline{m}_i(x_i,t)(\gamma)] \stackrel{\text{def}}{=} \{s : m_i(x_i,t,s) \ge \gamma\}.$$

- Then, for each i, β, and γ, we compute:

$$[\underline{m}_i^-(x_i)(\beta,\gamma),\underline{m}_i^+(x_i)(\beta,\gamma)] \stackrel{\text{def}}{=} \{t : \underline{m}_i(x_i,t)(\gamma) \ge \beta\} \text{ and}$$

$$[\overline{m}_i^-(x_i)(\beta,\gamma),\overline{m}_i^+(x_i)(\beta,\gamma)] \stackrel{\text{def}}{=} \{t : \overline{m}_i(x_i,t)(\gamma) \ge \beta\}.$$

- Then, for each i, α, β, and γ, we compute

$$\underline{\mathbf{x}}_i^-(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{x_i : \underline{m}_i^-(x_i)(\beta,\gamma) \ge \alpha\},$$

$$\underline{\mathbf{x}}_i^+(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{x_i : \underline{m}_i^+(x_i)(\beta,\gamma) \ge \alpha\},$$

$$\overline{\mathbf{x}}_i^-(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{x_i : \overline{m}_i^-(x_i)(\beta,\gamma) \ge \alpha\},$$

$$\overline{\mathbf{x}}_i^+(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \{x_i : \overline{m}_i^+(x_i)(\beta,\gamma) \ge \alpha\}.$$

- For each α, β, and γ, we then use an interval computation algorithm to compute:

$$\underline{\mathbf{y}}^-(\alpha,\beta,\gamma) = f(\underline{\mathbf{x}}_1^-(\alpha,\beta,\gamma),\ldots,\underline{\mathbf{x}}_n^-(\alpha,\beta,\gamma)),$$
$$\underline{\mathbf{y}}^+(\alpha,\beta,\gamma) = f(\underline{\mathbf{x}}_1^+(\alpha,\beta,\gamma),\ldots,\underline{\mathbf{x}}_n^+(\alpha,\beta,\gamma)),$$
$$\overline{\mathbf{y}}^-(\alpha,\beta,\gamma) = f(\overline{\mathbf{x}}_1^-(\alpha,\beta,\gamma),\ldots,\overline{\mathbf{x}}_n^-(\alpha,\beta,\gamma)),$$
$$\overline{\mathbf{y}}^+(\alpha,\beta,\gamma) = f(\overline{\mathbf{x}}_1^+(\alpha,\beta,\gamma),\ldots,\overline{\mathbf{x}}_n^+(\alpha,\beta,\gamma)).$$

- Next, for each y, β, and γ, we compute

$$\underline{m}^-(y)(\beta,\gamma) = \max\{\alpha : y \in \underline{\mathbf{y}}^-(\alpha,\beta,\gamma)\},$$

$$\underline{m}^+(y)(\beta,\gamma) = \max\{\alpha : y \in \underline{\mathbf{y}}^+(\alpha,\beta,\gamma)\},$$

$$\overline{m}^-(y)(\beta,\gamma) = \max\{\alpha : y \in \overline{\mathbf{y}}^-(\alpha,\beta,\gamma)\},$$

$$\overline{m}^+(y)(\beta,\gamma) = \max\{\alpha : y \in \overline{\mathbf{y}}^+(\alpha,\beta,\gamma)\}.$$

- For each y, t, and γ, we compute

$$\underline{m}(y,t)(\gamma) = \max\{\beta : t \in [\underline{m}^-(y)(\beta,\gamma), \underline{m}^+(y)(\beta,\gamma)]\} \text{ and}$$

$$\overline{m}(y,t)(\gamma) = \max\{\beta : t \in [\overline{m}^-(y)(\beta,\gamma), \overline{m}^+(y)(\beta,\gamma)]\}.$$

- Finally, for all y, t and s, we compute

$$m(y,t,s) = \max\{\gamma : s \in [\underline{m}(y,t)(\gamma), \overline{m}(y,t)(\gamma)]\}.$$

What about higher order fuzzy sets? In this section, we showed how processing type-2 fuzzy information can be used to processing type-2 fuzzy information. This reduction was based on the fact that in the type-3 case, each value $m(y)$ and $m_i(x_i)$ is a type-2 fuzzy set. Thus, we have the relation (1) between these type-2 fuzzy sets.

Similarly, in the type-4 case, each value $m(y)$ and $m_i(x_i)$ is a type-3 fuzzy set. Thus, we have the relation (1) between these type-3 fuzzy sets – and we can use the above algorithm to process these values. Similarly, for every level L, in the type-L case, each value $m(y)$ and $m_i(x_i)$ is a type-$(L-1)$ fuzzy set. Thus, we have the relation (1) between these type-$(L-1)$ fuzzy sets. This way, we can reduce processing type-L fuzzy sets to processing type-$(L-1)$ fuzzy sets; similarly, we can reduce processing type-$(L-1)$ fuzzy sets to processing type-$(L-2)$ fuzzy sets, etc., until we get to the known algorithms for processing type-1 and type-2 fuzzy sets.

How many computational steps do we need. The only (minor) problem with processing type-3 and higher-order fuzzy sets is that as we go to higher and higher order, the computational complexity increases. Indeed:

- For type-1, for each y, the desired information $m(y)$ consists of a single number. In this case, if we use 11 values of α, we need to use an interval computation algorithm 11 times.

- For type-2, for each y, we need to find the values $m(y,t)$ corresponding to different values $t \in [0,1]$. If we use 11 values for t, we thus need at least 11 times more computations than in the type-1 case – and indeed, we need order of $11 \cdot 11$ calls to an interval computation algorithm – namely, $2 \cdot 11^2$ calls.

- For type-3, for each y, we need to find the values $m(y,t,s)$ corresponding to different values $t,s \in [0,1]$. If we use 11 values of each of the variables t and s, we thus need at least 11^2 times more computations than in the type-1 case – and indeed, we need order of $11^2 \cdot 11 == 11^3$ calls to an interval computation algorithm – namely, $2^2 \cdot 11^3$ calls.

- In general, for type-L, for each y, we need to find the values $m(y,t_1,\ldots,t_{L-1})$ corresponding to different values $t_1,\ldots,t_{L-1} \in [0,1]$. If we use 11 values for each of the variables t_i, we thus need at least 11^{L-1} times more computations than in the type-1 case – and indeed, as one can show by induction over L, we need order of $11^{L-1} \cdot 11 = 11^L$ calls to an interval computation algorithm – namely, $2^{L-1} \cdot 11^L$ calls.

Acknowledgments

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), and HRD-1834620 and HRD-2034030 (CAHSI Includes), and by the AT&T Fellowship in Information Technology.

It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478, and by a grant from the Hungarian National Research, Development and Innovation Office (NRDI).

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] O. Castillo, J. Castro, and P. Melin, *Interval Type-3 Fuzzy Systems: Theory and Design*, Springer, Cham, Switzerland, 2022.

[3] O. Castillo, J. Castro, and P. Melin, "A methodology for building interval type-3 fuzzy systems based on the principle of justifiable granularity", *International Journal of Intelligent Systems*, 2022, doi 10.1002/int.22910

[4] O. Castillo, J. Castro, and P. Melin, "Interval type-3 fuzzy aggregation of neural networks for multiple time series prediction: the case of financial forecasting", *Axioms*, 2022, Vol. 11, Paper 251.

[5] O. Castillo, J. Castro, and P. Melin, "Interval type-3 fuzzy control for automated tuning of image quality in televisions", *Axioms*, 2022, Vol. 11, Paper 276.

[6] L. Jaulin, M. Kiefer, O. Didrit, and E. Walter, *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control, and Robotics*, Springer, London, 2001.

[7] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[8] V. Kreinovich, "From processing interval-valued fuzzy data to general type-2: towards fast algorithms", *Proceedings of the IEEE Symposium on Advances in Type-2 Fuzzy Logic Systems T2FUZZ'2011, part of the IEEE Symposium Series on Computational Intelligence*, Paris, France, April 11–15, 2011, pp. ix–xii.

[9] V. Kreinovich and G. Xiang, "Towards fast algorithms for processing type-2 fuzzy data: extending Mendel's algorithms from interval-valued to a more general case", *Proceedings of the 27th International Conference of the North American Fuzzy Information Processing Society NAFIPS'2008*, New York, New York, May 19–22, 2008.

[10] B. J. Kubica, *Interval Methods for Solving Nonlinear Contraint Satisfaction, Optimization, and Similar Problems: from Inequalities Systems to Game Solutions*, Springer, Cham, Switzerland, 2019.

[11] G. Mayer, *Interval Analysis and Automatic Result Verification*, de Gruyter, Berlin, 2017.

[12] J. M. Mendel, *Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions*, Springer, Cham, Switzerland, 2017.

[13] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*, SIAM, Philadelphia, 2009.

[14] H. T. Nguyen, C. L. Walker, and E. A. Walker, *A First Course in Fuzzy Logic*, Chapman and Hall/CRC, Boca Raton, Florida, 2019.

[15] V. Novák, I. Perfilieva, and J. Močkoř, *Mathematical Principles of Fuzzy Logic*, Kluwer, Boston, Dordrecht, 1999.

[16] L. A. Zadeh, "Fuzzy sets", *Information and Control*, 1965, Vol. 8, pp. 338–353.