Towards Fast Algorithms for Fuzzy Data Processing: Type 1, Type 2, and Beyond or How Interval Ideas Travelled from Warszawa, Tokyo, and California Back to Warszawa

Vladik Kreinovich

Department of Computer Science
University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, vladik@utep.edu

Many of these results come from joint papers
with Andrzej Pownuk

1. Outline

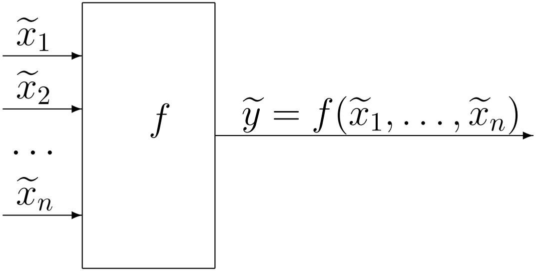
- From the mathematical viewpoint, we can use Zadeh's extension principle to process fuzzy data.
- However, a direct implementation of Zadeh's extension principle often requires too many computational steps.
- A known way to speed up computations is to use interval computations on α-cuts.
- We show that in many cases, we can further reduce computation time.
- The need to decrease computation time is even more important for type-2 fuzzy sets.
- They require even more computations; we show that for type-2, a significant speed up is also possible.
- We also extend the speed-up beyond the min t-norm.

Part I

Fuzzy Data Processing: What Is the Problem, and Which Algorithms Are Available for Solving This Problem

2. Why Data Processing and Knowledge Processing Are Needed in the First Place

- *Problem:* some quantities y are difficult (or impossible) to measure or estimate directly.
- *Solution:* indirect measurements or estimates



- *Fact:* estimates $\widetilde{x}_i$ are approximate.
- *Question:* how approximation errors $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$ affect the resulting error $\Delta y = \widetilde{y} - y$?

3. From Probabilistic to Interval Uncertainty

- Manufacturers of MI provide us with bounds Δ_i on measurement errors: $|\Delta x_i| \le \Delta_i$.
- Thus, we know that $x_i \in [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$.
- Often, we also know probabilities, but in 2 cases, we don't:
  - cutting-edge measurements;
  - cutting-cost manufacturing.
- In such situations:
  - we know the intervals $[\underline{x}_i, \overline{x}_i] = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$ of possible values of x_i, and
  - we want to find the range of possible values of y:

$$\mathbf{y} = [\underline{y}, \overline{y}] = \{f(x_1, \ldots, x_n) : x_1 \in [\underline{x}_1, \overline{x}_1], \ldots, [\underline{x}_n, \overline{x}_n]\}.$$

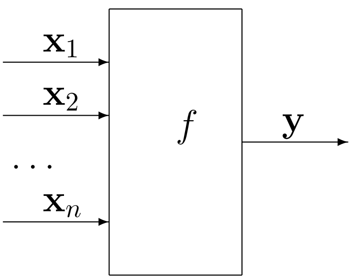
4. Main Problem of Interval Computations

We are given:

- an integer n;
- n intervals $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1]$, …, $\mathbf{x}_n = [\underline{x}_n, \overline{x}_n]$, and
- an algorithm $f(x_1, \ldots, x_n)$ which transforms n real numbers into a real number $y = f(x_1, \ldots, x_n)$.

We need to compute the endpoints $\underline{y}$ and $\overline{y}$ of the interval

$$\mathbf{y} = [\underline{y}, \overline{y}] = \{f(x_1, \ldots, x_n) : x_1 \in [\underline{x}_1, \overline{x}_1], \ldots, [\underline{x}_n, \overline{x}_n]\}.$$



Outline
Fuzzy Data Processing
Interval Computations
Reduction to Interval …
Measurement and …
Further Speed-Up
Type-2 Fuzzy Case
Beyond min t-Norm
Reduction to Informal …
Home Page
Title Page
◀◀ ▶▶
◀ ▶
Page 6 of 45
Go Back
Full Screen
Close
Quit

5. Interval Computations: A Brief History

- *Origins:* Archimedes (Ancient Greece)
- *Pioneers:* Mieczyslaw Warmus (Poland), Teruo Sunaga (Japan), Raymond Moore (USA), 1956–59
- *First boom:* early 1960s.
- *First challenge:* taking interval uncertainty into account when planning spaceflights to the Moon.
- *Current applications* (sample):
  - design of elementary particle colliders: Martin Berz, Kyoko Makino (USA)
  - will a comet hit the Earth: Berz, Moore (USA)
  - robotics: Jaulin (France), Neumaier (Austria)
  - chemical engineering: Mark Stadtherr (USA)

6. Need to Process Fuzzy Uncertainty

- In many practical situations, we only have expert estimates for the inputs x_i.
- Sometimes, experts provide guaranteed bounds on x_i, and even the probabilities of different values.
- However, such cases are rare.
- Usually, the experts' opinion is described by (imprecise, "fuzzy") words from natural language.
- Example: the value x_i of the i-th quantity is approximately 1.0, with an accuracy most probably about 0.1.
- Based on such "fuzzy" information, what can we say about $y = f(x_1, \ldots, x_n)$?
- The need to process such "fuzzy" information was first emphasized in the early 1960s by L. Zadeh.

7. How to Describe Fuzzy Uncertainty: Reminder

- In Zadeh's approach, we assign:
  - to each number x_i,
  - a degree $m_i(x_i) \in [0,1]$ with which x_i is a possible value of the i-th input.
- In most practical situations, the membership function:
  - starts with 0,
  - continuously $\uparrow$ until a certain value,
  - and then continuously $\downarrow$ to 0.
- Such membership function describe usual expert's expressions such as "small", "$\approx a$ with an error $\approx \sigma$".
- Membership functions of this type are actively used in expert estimates of number-valued quantities.
- They are thus called *fuzzy numbers*.

8. Processing Fuzzy Data: Formulation of the Problem

- We know an algorithm $y = f(x_1, \ldots, x_n)$ that relates:
  - the value of the desired difficult-to-estimate quantity y with
  - the values of easier-to-estimate auxiliary quantities $x_1, \ldots, x_n$.
- We also have expert knowledge about each of the quantities x_i.
- For each i, this knowledge is described in terms of the corresponding membership function $m_i(x_i)$.
- Based on this information, we want to find the membership function $m(y)$ which describes:
  - for each real number y,
  - the degree of confidence that this number is a possible value of the desired quantity.

9. Towards Solving the Problem

- Intuitively, y is a possible value of the desired quantity if for some values $x_1, \ldots, x_n$:
  - x_1 is a possible value of the 1st input quantity,
  - and x_2 is a possible value of the 2nd input quantity,
  - $\ldots$,
  - and $y = f(x_1 \ldots, x_n)$.
- We know:
  - that the degree of confidence that x_1 is a possible value of the 1st input quantity is equal to $m_1(x_1)$,
  - that the degree of confidence that x_2 is a possible value of the 2nd input quantity is equal to $m_2(x_2)$,
  - etc.
- The degree of confidence $d(y, x_1, \ldots, x_n)$ in an equality $y = f(x_1 \ldots, x_n)$ is, of course, 1 or 0.

10. Towards Solving the Problem (cont-d)

- The simplest way to represent "and" is to use min.
- Thus, for each combination of values $x_1, \ldots, x_n$, the degree of confidence d in a composite statement

  "x_1 is a possible value of the 1st input quantity, and x_2 is a possible value of the 2nd input quantity, ..., and $y = f(x_1 \ldots, x_n)$"

  is equal to

  $$d = \min(m_1(x_1), m_2(x_2), \ldots, d(y, x_1, \ldots, x_n)).$$

- We can simplify this expression if we consider two possible cases:
  - when $y = f(x_1 \ldots, x_n)$, we get

    $$d = \min(m_1(x_1), m_2(x_2), \ldots, d(y, x_1, \ldots, x_n));$$

  - otherwise, we get $d = 0$.

11. Using "Or"

- We want to combine these degrees of belief into a single degree of confidence that for some values $x_1, \ldots, x_n$,
  - x_1 is a possible value of the 1st input quantity,
  - and x_2 is a possible value of the 2nd quantity, ...,
  - and $y = f(x_1 \ldots, x_n)$.
- The words "for some values $x_1, \ldots, x_n$" means that the following composite property hold
  - either for one combination of real numbers $x_1, \ldots, x_n$,
  - or from another combination, etc.
- The simplest way to represent "or" is to use max.
- Thus, the desired degree of confidence $m(y)$ is equal to the maximum of the degrees corr. to different x_i:

$$m(y) = \sup_{x_1,\ldots,x_n} \min(m_1(x_1), m_2(x_2), \ldots, d(y, x_1, \ldots, x_n)).$$

12. Zadeh's Extension Principle

- $m(y) = \sup\limits_{x_1,\ldots,x_n} \min(m_1(x_1), m_2(x_2), \ldots, d(y, x_1, \ldots, x_n)).$
- We know that the maximized degree is non-zero only when $y = f(x_1 \ldots, x_n)$.
- It is therefore sufficient to only take supremum over such combinations.
- For such combinations, we can omit the term $d(y, x_1, \ldots, x_n)$ in the maximized expression.
- So, we arrive at the following formula:

  $$m(y) = \sup\{\min(m_1(x_1), m_2(x_2), \ldots) : y = f(x_1, \ldots, x_n)\}.$$

- This formula was first proposed by L. Zadeh and is thus called *Zadeh's extension principle.*
- This is the main formula that describes knowledge processing under fuzzy uncertainty.

13. Reduction to Interval Computations

- $m(y) = \sup\{\min(m_1(x_1), \ldots) : y = f(x_1, \ldots, x_n)\}$.
- Knowledge processing under fuzzy uncertainty is usually done by reducing to interval computations.
- Specifically, for each fuzzy set $m(x)$ and for each α in $(0, 1]$, we can define its α-*cut* $\mathbf{x}(\alpha) \stackrel{\text{def}}{=} \{x : m(x) \ge \alpha\}$.
- Vice versa, if we know the α-cuts for all α, we can reconstruct $m(x)$ as the largest α for which $x \in \mathbf{x}(\alpha)$.
- When $m_i(x_i)$ are fuzzy numbers, and $y = f(x_1, \ldots, x_n)$ is continuous, then for each α, we have:

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

- There exist many efficient algorithms and software packages for solving interval computations problems.
- So, the above reduction can help to efficiently solve the problems of fuzzy data processing as well.

14. Measurement and Estimation Inaccuracies Are Usually Small

- In many practical situations, the measurement and estimation inaccuracies Δx_i are relatively small.
- Then, we can safely ignore terms which are quadratic (or of higher order) in terms of Δx_i:

$$\Delta y = \widetilde{y} - y = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n) = \sum_{i=1}^{n} c_i \cdot \Delta x_i, \text{ where } c_i = \frac{\partial f}{\partial x_i}.$$

- If needed, the derivative can be estimated by numerical differentiation

$$c_i \approx \frac{f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}}{h}.$$

15. Estimating Accuracy of Data Processing

- The value $\Delta y = \sum\limits_{i=1}^{n} c_i \cdot \Delta x_i$ is the largest when each term is the largest, so $\Delta = \sum\limits_{i=1}^{n} |c_i| \cdot \Delta_i$.
- In the fuzzy case, the similar formula holds for the α-cuts, for every α: ${}^{\alpha}\Delta = \sum\limits_{i=1}^{n} |c_i| \cdot {}^{\alpha}\Delta_i$.
- Experts cannot describe their degrees of confidence α with too much accuracy.
- Usually, it is sufficient to consider only eleven values $\alpha = 0.0$, $\alpha = 0.1$, $\alpha = 0.2$, $\ldots$, $\alpha = 0.9$, and $\alpha = 1.0$.
- Thus, we need to apply the above formula eleven times.
- This is in line with the fact we usually divide each quantity into 7 ± 2 categories (Miller's "7 ± 2 Law").
- So, it is sufficient to have at least 9 different categories.

Part II

In Many Practical Situations, We Can Further Speed Up Fuzzy Data Processing

16. When a Further Speed-Up Is Possible?

- Sometimes, all membership functions are "of the same type": $m(z) = m_0(k \cdot z)$ for some symmetric $m_0(z)$.
- *Example:* for triangular functions,

$$m_0(z) = \max(1 - |z|, 0).$$

- In this case, $m(z) \geq \alpha$ is equivalent to $m_0(k \cdot z) \geq \alpha$, so ${}^\alpha\Delta_0 = k \cdot {}^\alpha\Delta$ and ${}^0\Delta_0 = k \cdot {}^0\Delta$.
- Thus, ${}^\alpha\Delta = f(\alpha) \cdot {}^0\Delta$, where $f(\alpha) \stackrel{\text{def}}{=} \dfrac{{}^\alpha\Delta_0}{{}^0\Delta_0}$.
- For example, for a triangular membership function, we have $f(\alpha) = 1 - \alpha$.
- So, if we know the type m_0 (hence $f(\alpha)$), and we know the 0-cut, we can compute all α-cuts as ${}^\alpha\Delta = f(\alpha) \cdot {}^0\Delta$.
- So, if $m_i(\Delta x_i)$ are of the same type, then ${}^\alpha\Delta_i = f(\alpha) \cdot {}^0\Delta_i$ for all α.

17. When a Speed-Up Is Possible (cont-d)

- We know that ${}^\alpha\Delta = \sum\limits_{i=1}^n |c_i| \cdot {}^\alpha\Delta_i$.
- For ${}^\alpha\Delta_i = f(\alpha) \cdot {}^0\Delta_i$, we get

$${}^\alpha\Delta = \sum_{i=1}^n |c_i| \cdot f(\alpha) \cdot {}^0\Delta_i.$$

- So, ${}^\alpha\Delta = f(\alpha) \cdot \left(\sum\limits_{i=1}^n |c_i| \cdot {}^0\Delta_i \right) = f(\alpha) \cdot {}^0\Delta$.
- Thus, if all $m(x)$ are of the same type $m_0(z)$, there is no need to compute ${}^\alpha\Delta$ eleven times:
  – it is sufficient to compute ${}^0\Delta$;
  – to find all other values ${}^\alpha\Delta$, we simply multiply ${}^0\Delta$ by the factors $f(\alpha)$ corresponding to $m_0(z)$.

18. A More General Case

- A more general case is:
  - when we have a list of T different types of uncertainty – i.e., types of membership functions, and
  - each approximation error Δx_i consists of $\le T$ components of the corresponding type t:

$$\Delta x_i = \sum_{t=1}^{T} \Delta x_{i,t}.$$

- For example:
  - type $t = 1$ may correspond to intervals (which are, of course, a particular case of fuzzy uncertainty),
  - type $t = 2$ may correspond to triangular membership functions, etc.

19. How This Case Is Processed Now

- *First stage:*
  - we use the known membership functions $m_{i,t}(\Delta x_{i,t})$
  - to find the memberships functions $m_i(\Delta x_i)$ that correspond to the sums Δx_i.
- *Second stage:* we use $m_i(\Delta x_i)$ to compute the desired membership function $m(\Delta y)$.
- *Problem:* on the second stage, we apply the above formula eleven times:

$$^\alpha\Delta = \sum_{i=1}^{n} |c_i| \cdot {}^\alpha\Delta_i.$$

20. The Main Idea of this Section

- We have $\Delta y = \sum\limits_{i=1}^{n} c_i \cdot \Delta x_i$, where

$$\Delta x_i = \sum_{t=1}^{T} \Delta x_{i,t}.$$

- Thus, $\Delta y = \sum\limits_{i=1}^{n} c_i \cdot \left(\sum\limits_{t=1}^{T} \Delta x_{i,t}\right)$.
- Grouping together all the terms corr. to type t, we get $\Delta y = \sum\limits_{t=1}^{T} \Delta y_t$, where $\Delta y_t \stackrel{\text{def}}{=} \sum\limits_{i=1}^{n} c_i \cdot \Delta x_{i,t}$.
- For each t, we are combining membership functions of the same type, so it is enough to compute ${}^0\Delta_t$.
- Then, we add the resulting membership functions – by adding the corresponding α-cuts.

21. Resulting Algorithm

- Let $[-{}^0\Delta_{i,t}, {}^0\Delta_{i,t}]$ be 0-cuts of the membership functions $m_{i,t}(\Delta x_{i,t})$.
- Based on these 0-cuts, we compute, for each type t, the values ${}^0\Delta = \sum\limits_{i=1}^{n} |c_i| \cdot {}^0\Delta_{i,t}$.
- Then, for $\alpha = 0$, $\alpha = 0.1$, $\ldots$, and for $\alpha = 1.0$, we compute the values ${}^\alpha\Delta_t = f_t(\alpha) \cdot {}^0\Delta_t$.
- Finally, we add up α-cuts corresponding to different types t, to come up with the expression ${}^\alpha\Delta = \sum\limits_{t=1}^{T} {}^\alpha\Delta_t$.
- *Comment.* We can combine the last two steps into a single step: ${}^\alpha\Delta = \sum\limits_{t=1}^{T} f_t(\alpha) \cdot {}^0\Delta_t$.

22. The New Algorithm Is Much Faster

- The original algorithm computed the above formula eleven times:
$$^{\alpha}\Delta = \sum_{i=1}^{n} |c_i| \cdot {}^{\alpha}\Delta_i.$$
- The new algorithm uses the corresponding formula T times, i.e., as many times as there are types.
- All the other computations are much faster, since they do not grow with the input size n.
- Thus, if the number T of different types is smaller than eleven, the new methods is much faster.
- *Example:* for $T = 2$ types (e.g., intervals and triangular $m(x)$), we get a $\dfrac{11}{2} = 5.5$ times speedup.

Part III
Type-2 Fuzzy Case: What Is Known and How to Further Speed Up Data Processing

23. Need for Type-2 Fuzzy Sets

- Fuzzy logic analyzes cases when an expert cannot describe his/her knowledge by an exact value.
- Instead, the expert describe this knowledge by using words from natural language.
- Fuzzy logic described these words in a computer understandable form – as fuzzy sets.
- In the traditional approach to fuzzy logic, the expert's degree of certainty $m_A(x)$ is a number from $[0,1]$.
- However, we consider situations when an expert cannot describe his/her knowledge by a number.
- It is not reasonable to expect that the same expert will express his/her degree of certainty by an exact number.
- It is more reasonable to expect that the expert will describe $m(x)$ also by words from natural language.

24. Type-2 Fuzzy Sets

- It is reasonable to that the expert will describe these degrees also by words from natural language.
- Thus, a natural representation of the degree $m(x)$ is not a number, but rather a new fuzzy set.
- Such situations, in which to every value x we assign a fuzzy number $m(x)$, are called *type-2* fuzzy sets.
- Type-2 fuzzy sets provide a more adequate representation of expert knowledge.
- It is thus not surprising that in comparison with the more traditional type-1 sets, such sets lead to
  - a higher quality control,
  - higher quality clustering, etc.
- If type-2 fuzzy sets are more adequate, why are not they used more?

25. The Main Obstacle to Using Type-2 Fuzzy Sets

- Main reason: transition to type-2 fuzzy sets leads to an increase in computation time.
- Indeed, to describe a traditional (type-1) membership function function, it is sufficient to describe,
  - for each value x,
  - a single number $m(x)$.
- In contrast, to describe a type-2 set,
  - for each value x,
  - we must describe the entire membership function – which needs several parameters to describe.
- We need more numbers just to store such information.
- So, we need more computational time to process all the numbers representing these sets.

26. Interval-Valued Fuzzy Sets

- In line with this reasoning:
  - the most widely used type-2 fuzzy sets are
  - the ones which require the smallest number of parameters to store.
- We are talking about *interval-valued* fuzzy numbers, in which:
  - for each x,
  - the degree of certainty $m(x)$ is an interval

$$\mathbf{m}(x) = [\underline{m}(x), \overline{m}(x)].$$

- To store each interval, we need exactly two numbers.
- This is the smallest possible increase over the single number needed to store the type-1 value $m(x)$.

27. Towards Fast Algorithms for Processing Interval-Valued Fuzzy Data (Mendel et al.)

- For interval-valued fuzzy data, we only know the interval $\mathbf{m}_i(x_i) = [\underline{m}_i(x), \overline{m}_i(x)]$ of possible values of $m_i(x_i)$.
- By applying Zadeh's extension principle to different $m_i(x_i) \in [\underline{m}_i(x), \overline{m}_i(x)]$, we get different values of

  $$m(y) = \sup\{\min(m_1(x_1), m_2(x_2), \ldots) : y = f(x_1, \ldots, x_n)\}.$$
- When the values $m_i(x_i)$ continuously change, the value $m(y)$ also continuously changes.
- We want to know the set of possible values of $m(y)$.
- So, for every y, the set $\mathbf{m}(y)$ of all possible values of $m(y)$ is an interval:

  $$\mathbf{m}(y) = [\underline{m}(y), \overline{m}(y)].$$
- Thus, to describe this set, it is sufficient, for each y, to describe the endpoints $\underline{m}(y)$ and $\overline{m}(y)$.

28. Towards Fast Algorithms for Processing Interval-Valued Fuzzy Data (cont-d)

- We want to compute the range of

$$m(y) = \sup\{\min(m_1(x_1), m_2(x_2), \ldots) : y = f(x_1, \ldots, x_n)\}.$$

- This expression is non-strictly increasing in $m_i(x_i)$, so:
  - $m(y)$ attains its smallest value when all the inputs $m_i(x_i)$ are the smallest:

$$\underline{m}(y) = \sup\{\min(\underline{m}_1(x_1), \underline{m}_2(x_2), \ldots) : y = f(x_1, \ldots, x_n)\};$$

  - $m(y)$ attains its largest value when all the inputs $m_i(x_i)$ are the largest:

$$\overline{m}(y) = \sup\{\min(\overline{m}_1(x_1), \overline{m}_2(x_2), \ldots) : y = f(x_1, \ldots, x_n)\}.$$

- So, we need to apply Zadeh's extension principle to lower and membership functions $\underline{m}_i(x_i)$ and $\overline{m}_i(x_i)$.

29. Fast Algorithms for Processing Interval-Valued Fuzzy Data

- To find $\underline{m}(y)$ (corr., $\overline{m}(y)$), we apply Zadeh's extension principle to membership f-s $\underline{m}_i(x_i)$ (corr., $\overline{m}_i(x_i)$).
- For type-1 fuzzy sets, Zadeh's extension principle can be reduced to interval computations.
- Let $\underline{\mathbf{y}}(\alpha)$ denote α-cuts for $\underline{m}(y)$, and let $\overline{\mathbf{y}}(\alpha)$ denote α-cuts for $\overline{m}(y)$.
- Then, we arrive at the following algorithm: for every $\alpha \in (0,1]$,
  - first compute

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

  - then compute

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

30. Extension to General Type-2 Fuzzy Numbers

- *Reminder:* Zadeh's extension principle

  $$m(y) = \sup\{\min(m_1(x_1), m_2(x_2), \ldots) : y = f(x_1, \ldots, x_n)\}.$$

- *General type-2 case:* $m_i(x_i)$ are fuzzy numbers, with β-cuts $(m_i(x_i))(\beta) = [\underline{(m_i(x_i))(\beta)}, \overline{(m_i(x_i))(\beta)}]$.

- Due to known relation with interval computations:

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

- Due to monotonicity:

$$\underline{(m(y))(\beta)} = \sup\{\min(\underline{(m_1(x_1))(\beta)}, \ldots) : y = f(x_1, \ldots, x_n)\};$$

$$\overline{(m(y))(\beta)} = \sup\{\min(\overline{(m_1(x_1))(\beta)}, \ldots) : y = f(x_1, \ldots, x_n)\}.$$

- Due to known relation with interval computations:

$$\underline{\mathbf{y}}(\alpha, \beta) = f(\underline{\mathbf{x}}_1(\alpha, \beta), \ldots);\ \ \overline{\mathbf{y}}(\alpha, \beta) = f(\overline{\mathbf{x}}_1(\alpha, \beta), \ldots),\ \text{where}$$

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

31. Conclusions of this Section

- Type-2 fuzzy sets more adequately describe expert's opinion than the more traditional type-1 fuzzy sets.
- The use of type-2 fuzzy sets has thus led to better quality control, better quality clustering, etc.
- *Main obstacle:* the computational time of data processing increases.
- *Known result:* processing *interval-valued* fuzzy numbers can be reduced to interval computations.
- *Conclusion:* processing interval-valued fuzzy data is (almost) as fast as processing type-1 fuzzy data.
- In this talk, we showed that fast algorithms can be extended to *general* type-2 fuzzy numbers.
- This will hopefully lead to more practical applications of type-2 fuzzy sets.

Part IV
Beyond min t-Norms

32. Need to Go Beyond min t-Norm

- Usually, Zadeh's extension principle is applied to the situations in which we use the min "and"-operation

$$f_\&(a, b) = \min(a, b).$$

- However, in many practical situations, other "and"-operations more adequately describe expert reasoning.
- It is therefore desirable to consider the general case

$$m(y) = \sup\{f_\&(m_1(x_1), m_2(x_2), \ldots : u = f(x_1, \ldots, x_n)\}.$$

- We are interested in the linearized case, when

$$\Delta y = \sum_{i=1}^{n} c_i \cdot \Delta x_i.$$

- How can we speed up computations in this general case?

33. We Can Reduce This Problem to Computing the Sum of Two Fuzzy Quantities

- We know that $\Delta y = \sum\limits_{i=1}^{n} y_i$, where $y_i \stackrel{\text{def}}{=} c_i \cdot \Delta x_i$.
- Once we know the membership f-n $m_i(\Delta x_i)$, we can easily find the membership f-n for y_i as $m_i(y_i/c_i)$.
- If we know how to find the membership f-n for the sum of two fuzzy quantities, then we can:
  - find the membership f-n for $s_2 = y_1 + y_2$;
  - then, find membership f-n for
$$s_3 = s_2 + y_2 \ (= y_1 + y_2 + y_3),$$
  - etc, until we reach $s_n = y$.
- For fuzzy quantities with membership f-ns $n_1(x_1)$ and $n_2(x_2)$, the membership f-n for $y = x_1 + x_2$ is
$$n(y) = \max_{x_1} f_{\&}(n_1(x_1), n_2(y - x_1)).$$

34. Reduction to the Case of Product t-Norm

- We want to compute

$$n(y) = \max_{x_1} f_\&(n_1(x_1), n_2(y - x_1)).$$

- Some t-norms are *Archimedean*; for them:

$$f_\&(a, b) = f^{-1}(f(a) \cdot f(b)) \text{ for some function } f(x).$$

- Archimedean t-norms are universal approximators:
  - for every $\varepsilon > 0$,
  - every t-norm $f_\&$ is ε-close to some Archimedean.
- Thus, from the practical viewpoint, we can safely assume that $f_\&$ is Archimedean.
- Then, for $q_i(x_i) \stackrel{\text{def}}{=} f(n_i(x_i))$ and $q(y) \stackrel{\text{def}}{=} f(n(y))$:

$$q(y) = \max_{x_1} q_1(x_1) \cdot q_2(y - x_1).$$

35. Reduction to Informal Convolution

- We want to compute $q(y) = \max\limits_{x_1} q_1(x_1) \cdot q_2(y - x_1)$.
- For $\ell_i(x_i) \stackrel{\text{def}}{=} -\ln(q_i(x_i))$ and $\ell(y) \stackrel{\text{def}}{=} -\ln(q(y))$:
$$\ell(y) = \min_{x_1}(\ell_1(x_1) + \ell_2(y - x_1)).$$
- This operation is known as *infimal convlution*, and denoted by $\ell = \ell_1 \,\Box\, \ell_2$.
- It is known that $\ell_1 \Box \ell_2 = (\ell_1^* + \ell_2^*)^*$ for *Legendre transform*
$$\ell^*(s) \stackrel{\text{def}}{=} \sup_x (s \cdot x - \ell(x)).$$
- There exists a linear-time algorithm for computing Legendra transform.
- Thus, we can compute $\ell_1 \,\Box\, \ell_2$ in linear time.

36. Resulting Linear Time Algorithm for Fuzzy Data Processing Beyond min t-Norm

- We are given:
  - a function $f(x_1, \ldots, x_n)$;
  - n membership functions $m_1(x_1)$, ..., $m_n(x_n)$; and
  - an "and"-operation $f_\&(a, b)$.
- We want to compute a new membership function

$$m(y) = \max\{f_\&(m_1(x_1), \ldots, m_n(x_n)) : f(x_1, \ldots, x_n) = y\}.$$

- First, we represent $f_\&$ in the Acrhiemdean form $f_\&(a, b) = f^{-1}(f(a) \cdot f(b))$ for an appropriate $f(x)$.
- We thus assume that we have algorithms for computing $f(x)$ and the inverse function $f^{-1}(x)$.
- Then, for each i, we find the value $\widetilde{x}_i$ for which $m_i(x_i)$ attains its largest possible value $m_i(\widetilde{x}_i) = 1$.

37. Algorithm (cont-d)

- We then compute the values $c_0 = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$, $c_i = \dfrac{\partial f}{\partial x_i}_{|x_i=\widetilde{x}_i}$, and $a_0 = c_0 - \sum\limits_{i=1}^{n} c_i \cdot \widetilde{x}_i$.
- Then, $f(x_1, \ldots, x_n) \approx a_0 + \sum\limits_{i=1}^{n} c_i \cdot x_i$.
- After that, e compute the membership f-ns $n_1(s_1) = m_1((s_1 - a_0)/c_1)$ and $n_i(y_i) = m_i(y_i/c_i)$ for $i > 2$.
- In terms of the variables $s_1 = a_0 + c_1 \cdot x_1$ and $y_i = c_i \cdot x_i$, the desired quantity y has the form $y = s_1 + y_2 + \ldots + y_n$.
- We compute the minus logarithms of the resulting functions: $\ell_i(y_i) = -\ln(n_i(y_i))$.
- For each i, we then use the Fast Legendre Transform algorithm to compute ℓ_i^*.

38. Algorithm (final)

- Then, we add all these Legendre transforms and apply the Fast Legendre Transform once again, getting:

$$\ell = (\ell_1^* + \ldots + \ell_n^*)^*.$$

- This function $\ell(y)$ is equal to $\ell(y) = -\ln(n(y))$, so we can reconstruct $\nu(y)$ as $n(y) = \exp(-\ell(y))$.
- Finally, we can compute the desired membership function $m(u)$ as $m(y) = f^{-1}(n(y))$.

39. Conclusion for this Section

- To process fuzzy data, we need to use Zadeh's extension principle.
- In principle, this principle can be used for any t-norm.
- However, usually, it is only used for the min t-norm.
- Reason: only for this t-norm, an efficient (linear-time) algorithm for fuzzy data processing was known.
- In many practical situations, other t-norms are more adequate in describing expert's reasoning.
- We have shown that similar efficient linear-time algorithms can be designed for an arbitrary t-norm.
- Thus, it is possible to use a more adequate t-norm – and keep fuzzy data processing efficient.

40. Acknowledgments

- Many thanks for the organizers of EUSFLAT'2017 conference for the opportunity:
  - to present these results, and
  - to visit the beautiful Warszawa, one of the three birthplaces of interval computations.
- This work was supported in part by the National Science Foundation grant HRD-1242122.
- This grant supports the Cyber-ShARE Center of Excellence.