Os Lusíadas of Computations under Uncertainty: from Probabilities to Intervals to Fuzzy to Interval-Valued Fuzzy and Beyond

Vladik Kreinovich

Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
Email: vladik@utep.edu
http://www.cs.utep.edu/vladik

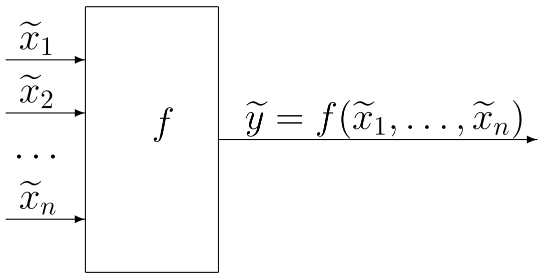
http://www.cs.utep.edu/interval-comp

1. Poetic Introduction

- There is a lot of uncertainty in our knowledge.
- In the glorious past, explorers sailed into the unknown seas and brought forth new knowledge.
- As a result of their efforts, the whole Earth has been thoroughly mapped.
- However, there are many areas which are as uncertain as the unknown lands were in the old days.
- For exploring the microworlds of cells and atoms, the macroworlds of galaxies, our "ships" are computers.
- Data processing under uncertainty – this is how we bring new knowledge about our world.
- We will try to show that data processing under uncertainty can be as exciting as sea voyages of yore.

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:

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

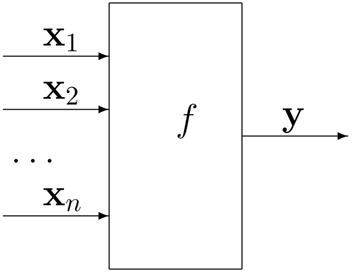
4. Main Problem of Interval Computations

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

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



5. 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.

6. 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*.

7. 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.

8. 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.

9. 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, $\ldots$, 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 $d = 0$.

10. 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:

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

11. 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.

12. Reduction to Interval Computations

- $m(y) = \sup\{\min(m_1(x_1), m_2(x_2), \ldots) : y = f(x_1, \ldots, x_n)\}$.
- Knowledge processing under fuzzy uncertainty is usually done by reducing to interval computations.
- Specifically, for each fuzzy set $m(x)$ and for each $\alpha \in (0, 1]$, we can define its α-*cut* $\mathbf{x}(\alpha) \stackrel{\text{def}}{=} \{x : m(x) \geq \alpha\}$.
- 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.

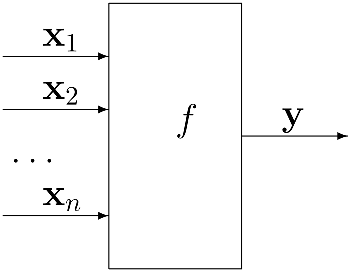
13. Main Problem of Interval Computations

We are given:

- an integer n;
- n intervals $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1]$, $\ldots$, $\mathbf{x}_n = [\underline{x}_n, \overline{x}_n]$, 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

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



14. Interval Arithmetic: Foundations of Interval Techniques

- *Problem:* compute the range

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

- *Interval arithmetic:* for arithmetic operations $f(x_1, x_2)$ (and for elementary functions), we have explicit formulas for the range.

- *Examples:* when $x_1 \in \mathbf{x}_1 = [\underline{x}_1, \overline{x}_1]$ and $x_2 \in \mathbf{x}_2 = [\underline{x}_2, \overline{x}_2]$, then:
  - The range $\mathbf{x}_1 + \mathbf{x}_2$ for $x_1 + x_2$ is $[\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2]$.
  - The range $\mathbf{x}_1 - \mathbf{x}_2$ for $x_1 - x_2$ is $[\underline{x}_1 - \overline{x}_2, \overline{x}_1 - \underline{x}_2]$.
  - The range $\mathbf{x}_1 \cdot \mathbf{x}_2$ for $x_1 \cdot x_2$ is $[\underline{y}, \overline{y}]$, where

    $$\underline{y} = \min(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2);$$

    $$\overline{y} = \max(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2).$$

- The range $1/\mathbf{x}_1$ for $1/x_1$ is $[1/\overline{x}_1, 1/\underline{x}_1]$ (if $0 \notin \mathbf{x}_1$).

15. Straightforward Interval Computations: Example

- *Example:* $f(x) = (x-2) \cdot (x+2)$, $x \in [1, 2]$.
- How will the computer compute it?
  - $r_1 := x - 2$;
  - $r_2 := x + 2$;
  - $r_3 := r_1 \cdot r_2$.
- *Main idea:* perform the same operations, but with *intervals* instead of *numbers*:
  - $\mathbf{r}_1 := [1, 2] - [2, 2] = [-1, 0]$;
  - $\mathbf{r}_2 := [1, 2] + [2, 2] = [3, 4]$;
  - $\mathbf{r}_3 := [-1, 0] \cdot [3, 4] = [-4, 0]$.
- *Actual range:* $f(\mathbf{x}) = [-3, 0]$.
- *Comment:* this is just a toy example, there are more efficient ways of computing an enclosure $\mathbf{Y} \supseteq \mathbf{y}$.

16. First Idea: Use of Monotonicity

- *Reminder:* for arithmetic, we had exact ranges.
- *Reason:* $+$, $-$, $\cdot$ are monotonic in each variable.
- *How monotonicity helps:* if $f(x_1, \ldots, x_n)$ is (non-strictly) increasing ($f \uparrow$) in each x_i, then
$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = [f(\underline{x}_1, \ldots, \underline{x}_n), f(\overline{x}_1, \ldots, \overline{x}_n)].$$
- *Similarly:* if $f \uparrow$ for some x_i and $f \downarrow$ for other x_j.
- *Fact:* $f \uparrow$ in x_i if $\dfrac{\partial f}{\partial x_i} \geq 0$.
- *Checking monotonicity:* check that the range $[\underline{r}_i, \overline{r}_i]$ of $\dfrac{\partial f}{\partial x_i}$ on $\mathbf{x}_i$ has $\underline{r}_i \geq 0$.
- *Differentiation:* by Automatic Differentiation (AD) tools.
- *Estimating ranges of* $\dfrac{\partial f}{\partial x_i}$*:* straightforward interval comp.

17. Monotonicity: Example

- *Idea:* if the range $[\underline{r}_i, \overline{r}_i]$ of each $\dfrac{\partial f}{\partial x_i}$ on $\mathbf{x}_i$ has $\underline{r}_i \ge 0$, then

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

- *Example:* $f(x) = (x-2) \cdot (x+2)$, $\mathbf{x} = [1, 2]$.
- *Case $n = 1$:* if the range $[\underline{r}, \overline{r}]$ of $\dfrac{df}{dx}$ on $\mathbf{x}$ has $\underline{r} \ge 0$, then

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

- *AD:* $\dfrac{df}{dx} = 1 \cdot (x+2) + (x-2) \cdot 1 = 2x$.
- *Checking:* $[\underline{r}, \overline{r}] = [2, 4]$, with $2 \ge 0$.
- *Result:* $f([1, 2]) = [f(1), f(2)] = [-3, 0]$.
- *Comparison:* this is the exact range.

18. Non-Monotonic Example

- *Example:* $f(x) = x \cdot (1 - x)$, $x \in [0, 1]$.
- How will the computer compute it?
  - $r_1 := 1 - x$;
  - $r_2 := x \cdot r_1$.
- *Straightforward interval computations:*
  - $\mathbf{r}_1 := [1, 1] - [0, 1] = [0, 1]$;
  - $\mathbf{r}_2 := [0, 1] \cdot [0, 1] = [0, 1]$.
- *Actual range:* min, max of f at $\underline{x}$, $\overline{x}$, or when $\dfrac{df}{dx} = 0$.
- Here, $\dfrac{df}{dx} = 1 - 2x = 0$ for $x = 0.5$, so
  - compute $f(0) = 0$, $f(0.5) = 0.25$, and $f(1) = 0$.
  - $\underline{y} = \min(0, 0.25, 0) = 0$, $\overline{y} = \max(0, 0.25, 0) = 0.25$.
- *Resulting range:* $f(\mathbf{x}) = [0, 0.25]$.

19. Second Idea: Centered Form

- *Main idea:* Intermediate Value Theorem

$$f(x_1,\ldots,x_n)=f(\widetilde{x}_1,\ldots,\widetilde{x}_n)+\sum_{i=1}^{n}\frac{\partial f}{\partial x_i}(\chi)\cdot(x_i-\widetilde{x}_i)$$

  for some $\chi_i\in\mathbf{x}_i$.

- *Corollary:* $f(x_1,\ldots,x_n)\in\mathbf{Y}$, where

$$\mathbf{Y}=\widetilde{y}+\sum_{i=1}^{n}\frac{\partial f}{\partial x_i}(\mathbf{x}_1,\ldots,\mathbf{x}_n)\cdot[-\Delta_i,\Delta_i].$$

- *Differentiation:* by Automatic Differentiation (AD) tools.
- *Estimating the ranges of derivatives:*
  - if appropriate, by monotonicity, or
  - by straightforward interval computations, or
  - by centered form (more time but more accurate).

20. Centered Form: Example

- *General formula:*

$$\mathbf{Y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \cdot [-\Delta_i, \Delta_i].$$

- *Example:* $f(x) = x \cdot (1 - x)$, $\mathbf{x} = [0, 1]$.
- Here, $\mathbf{x} = [\widetilde{x} - \Delta, \widetilde{x} + \Delta]$, with $\widetilde{x} = 0.5$ and $\Delta = 0.5$.
- *Case $n = 1$:* $\mathbf{Y} = f(\widetilde{x}) + \dfrac{df}{dx}(\mathbf{x}) \cdot [-\Delta, \Delta]$.
- *AD:* $\dfrac{df}{dx} = 1 \cdot (1 - x) + x \cdot (-1) = 1 - 2x$.
- *Estimation:* we have $\dfrac{df}{dx}(\mathbf{x}) = 1 - 2 \cdot [0, 1] = [-1, 1]$.
- *Result:* $\mathbf{Y} = 0.5 \cdot (1 - 0.5) + [-1, 1] \cdot [-0.5, 0.5] = 0.25 + [-0.5, 0.5] = [-0.25, 0.75]$.
- *Comparison:* actual range $[0, 0.25]$, straightforward $[0, 1]$.

21. Third Idea: Bisection

- *Known:* accuracy $O(\Delta_i^2)$ of first order formula

$$f(x_1,\ldots,x_n) = f(\widetilde{x}_1,\ldots,\widetilde{x}_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\chi)\cdot(x_i-\widetilde{x}_i).$$

- *Idea:* if the intervals are too wide, we:
  - split one of them in half ($\Delta_i^2 \to \Delta_i^2/4$); and
  - take the union of the resulting ranges.
- *Example:* $f(x) = x\cdot(1-x)$, where $x \in \mathbf{x} = [0,1]$.
- *Split:* take $\mathbf{x}' = [0, 0.5]$ and $\mathbf{x}'' = [0.5, 1]$.
- *1st range:* $1 - 2\cdot\mathbf{x} = 1 - 2\cdot[0, 0.5] = [0, 1]$, so $f \uparrow$ and $f(\mathbf{x}') = [f(0), f(0.5)] = [0, 0.25]$.
- *2nd range:* $1 - 2\cdot\mathbf{x} = 1 - 2\cdot[0.5, 1] = [-1, 0]$, so $f \downarrow$ and $f(\mathbf{x}'') = [f(1), f(0.5)] = [0, 0.25]$.
- *Result:* $f(\mathbf{x}') \cup f(\mathbf{x}'') = [0, 0.25]$ – exact.

22. 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.

23. 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?

24. 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.

25. 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 $m(x)$.

26. 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 $\overline{m}(y)$.

27. 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 $\overline{m}_i(x_i)$.

28. 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

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

29. New Result: 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:

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

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

Home Page
Title Page
◀◀ ▶▶
◀ ▶

Go Back
Full Screen
Close
Quit

30. Pragmatic Conclusions

- 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.

31. Poetic Conclusions

- A seafarer of old:
  - would sail into the West
  - and by doing that would eventually sail back home.
- Similarly, computational techniques:
  - get modified to take into account for newer and newer types of uncertainty, but
  - eventually return to the basics: to techniques for processing simple intervals.

32. Acknowledgments

This work was supported in part:

- by the National Science Foundation grants HRD-0734825 and DUE-0926721 and
- by Grant 1 T36 GM078000-01 from the National Institutes of Health.

33. Alternative Approach: Affine Arithmetic

- *So far:* we compute the range of $x \cdot (1-x)$ by multiplying ranges of x and $1-x$.
- *We ignore:* that both factors depend on x and are, thus, dependent.
- *Idea:* for each intermediate result a, keep an explicit dependence on $\Delta x_i = \widetilde{x}_i - x_i$ (at least its linear terms).
- *Implementation:*

$$a = a_0 + \sum_{i=1}^{n} a_i \cdot \Delta x_i + [\underline{a}, \overline{a}].$$

- *We start:* with $x_i = \widetilde{x}_i - \Delta x_i$, i.e.,

$$\widetilde{x}_i + 0 \cdot \Delta x_1 + \ldots + 0 \cdot \Delta x_{i-1} + (-1) \cdot \Delta x_i + 0 \cdot \Delta x_{i+1} + \ldots + 0 \cdot \Delta x_n + [0, 0].$$

- *Description:* $a_0 = \widetilde{x}_i$, $a_i = -1$, $a_j = 0$ for $j \neq i$, and $[\underline{a}, \overline{a}] = [0, 0]$.

34. Affine Arithmetic: Operations

- *Representation:* $a = a_0 + \sum_{i=1}^{n} a_i \cdot \Delta x_i + [\underline{a}, \overline{a}]$.
- *Input:* $a = a_0 + \sum_{i=1}^{n} a_i \cdot \Delta x_i + \mathbf{a}$ and $b = b_0 + \sum_{i=1}^{n} b_i \cdot \Delta x_i + \mathbf{b}$.
- *Operations:* $c = a \otimes b$.
- *Addition:* $c_0 = a_0 + b_0$, $c_i = a_i + b_i$, $\mathbf{c} = \mathbf{a} + \mathbf{b}$.
- *Subtraction:* $c_0 = a_0 - b_0$, $c_i = a_i - b_i$, $\mathbf{c} = \mathbf{a} - \mathbf{b}$.
- *Multiplication:* $c_0 = a_0 \cdot b_0$, $c_i = a_0 \cdot b_i + b_0 \cdot a_i$,

$$\mathbf{c} = a_0 \cdot \mathbf{b} + b_0 \cdot \mathbf{a} + \sum_{i \neq j} a_i \cdot b_j \cdot [-\Delta_i, \Delta_i] \cdot [-\Delta_j, \Delta_j] +$$

$$\sum_i a_i \cdot b_i \cdot [-\Delta_i, \Delta_i]^2 +$$

$$\left(\sum_i a_i \cdot [-\Delta_i, \Delta_i] \right) \cdot \mathbf{b} + \left(\sum_i b_i \cdot [-\Delta_i, \Delta_i] \right) \cdot \mathbf{a} + \mathbf{a} \cdot \mathbf{b}.$$

35. Affine Arithmetic: Example

- *Example:* $f(x) = x \cdot (1 - x)$, $x \in [0, 1]$.
- Here, $n = 1$, $\widetilde{x} = 0.5$, and $\Delta = 0.5$.
- How will the computer compute it?
  - $r_1 := 1 - x$;
  - $r_2 := x \cdot r_1$.
- *Affine arithmetic:* we start with $x = 0.5 - \Delta x + [0, 0]$;
  - $\mathbf{r}_1 := 1 - (0.5 - \Delta x) = 0.5 + \Delta x$;
  - $\mathbf{r}_2 := (0.5 - \Delta x) \cdot (0.5 + \Delta x)$, i.e.,

$$\mathbf{r}_2 = 0.25 + 0 \cdot \Delta x - [-\Delta, \Delta]^2 = 0.25 + [-\Delta^2, 0].$$

- *Resulting range:* $\mathbf{y} = 0.25 + [-0.25, 0] = [0, 0.25]$.
- *Comparison:* this is the exact range.

36. Affine Arithmetic: Towards More Accurate Estimates

- *In our simple example:* we got the exact range.
- *In general:* range estimation is NP-hard.
- *Meaning:* a feasible (polynomial-time) algorithm will sometimes lead to excess width: $\mathbf{Y} \supset \mathbf{y}$.
- *Conclusion:* affine arithmetic may lead to excess width.
- *Question:* how to get more accurate estimates?
- *First idea:* bisection.
- *Second idea* (Taylor arithmetic):
  - *affine arithmetic:* $a = a_0 + \sum a_i \cdot \Delta x_i + \mathbf{a}$;
  - *meaning:* we keep linear terms in Δx_i;
  - *idea:* keep, e.g., quadratic terms

$$a = a_0 + \sum a_i \cdot \Delta x_i + \sum a_{ij} \cdot \Delta x_i \cdot \Delta x_j + \mathbf{a}.$$

37. Interval Computations vs. Affine Arithmetic: Comparative Analysis

- *Objective:* we want a method that computes a reasonable estimate for the range in reasonable time.
- *Conclusion – how to compare different methods:*
  - how accurate are the estimates, and
  - how fast we can compute them.
- *Accuracy:* affine arithmetic leads to more accurate ranges.
- *Computation time:*
  - *Interval arithmetic:* for each intermediate result a, we compute two values: endpoints $\underline{a}$ and $\overline{a}$ of $[\underline{a}, \overline{a}]$.
  - *Affine arithmetic:* for each a, we compute $n+3$ values:

$$a_0 \quad a_1, \ldots, a_n \quad \underline{a}, \overline{a}.$$

- *Conclusion:* affine arithmetic is $\sim n$ times slower.