# Statistical Data Processing under Interval Uncertainty: Algorithms and Computational Complexity

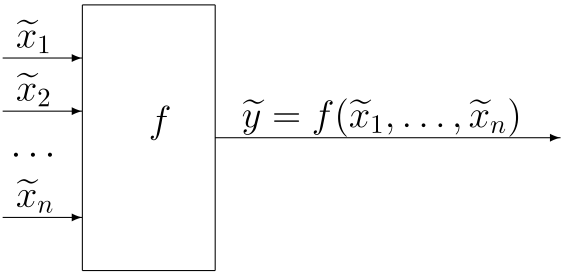
Vladik Kreinovich

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

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

## 1. General Problem of Data Processing under Uncertainty

- *Indirect measurements:* way to measure $y$ that are are difficult (or even impossible) to measure directly.
- *Idea:* $y = f(x_1, \ldots, x_n)$



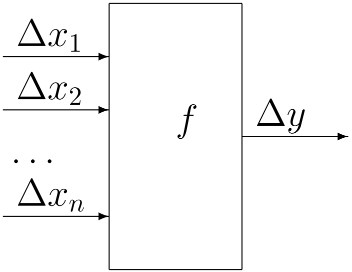

- *Problem:* measurements are never 100% accurate: $\widetilde{x}_i \neq x_i$ $(\Delta x_i \neq 0)$ hence

$$\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) \neq y = f(x_1, \ldots, y_n).$$

- *Question:* what are bounds on $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y$?


Title Page
◀◀ ▶▶
◀ ▶
Page 2 of 48
Go Back
Full Screen
Close
Quit

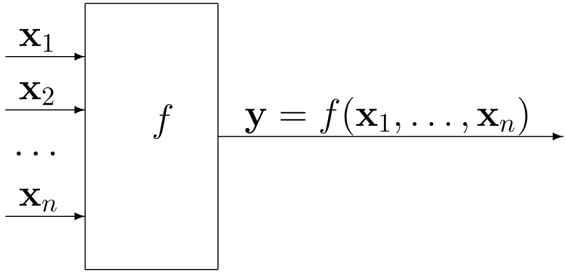
## 2. Probabilistic and Interval Uncertainty




- *Traditional approach:* we know probability distribution for $\Delta x_i$ (usually Gaussian).
- *Where it comes from:* calibration using standard MI.
- *Problem:* calibration is not possible in:
  - fundamental science
  - manufacturing
- *Solution:* we know upper bounds $\Delta_i$ on $|\Delta x_i|$ hence

$$x_i \in [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i].$$

# 3. Interval Computations: A Problem




- *Given:* an algorithm $y = f(x_1, \dots, x_n)$ and $n$ intervals $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$.
- *Compute:* the corresponding range of $y$:

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

- *Fact:* NP-hard even for quadratic $f$.
- *Challenge:* when are feasible algorithm possible?
- *Challenge:* when computing $\mathbf{y} = [\underline{y}, \overline{y}]$ is not feasible, find a good approximation $\mathbf{Y} \supseteq \mathbf{y}$.

# 4. Alternative Approach: Maximum Entropy

- *Situation:* in many practical applications, it is very difficult to come up with the probabilities.
- *Traditional engineering approach:* use probabilistic techniques.
- *Problem:* many different probability distributions are consistent with the same observations.
- *Solution:* select one of these distributions – e.g., the one with the largest entropy.
- *Example – single variable:* if all we know is that $x \in [\underline{x}, \overline{x}]$, then MaxEnt leads to a uniform distribution on $[\underline{x}, \overline{x}]$.
- *Example – multiple variables:* different variables are independently distributed.

## 5. Limitations of Maximum Entropy Approach

- *Example:* simplest algorithm $y = x_1 + \ldots + x_n$.
- *Measurement errors:* $\Delta x_i \in [-\Delta, \Delta]$.
- *Analysis:* $\Delta y = \Delta x_1 + \ldots + \Delta x_n$.
- *Worst case situation:* $\Delta y = n \cdot \Delta$.
- *Maximum Entropy approach:* due to Central Limit Theorem, $\Delta y$ is $\approx$ normal, with $\sigma = \Delta \cdot \dfrac{\sqrt{n}}{\sqrt{3}}$.
- *Why this may be inadequate:* we get $\Delta \sim \sqrt{n}$, but due to correlation, it is possible that $\Delta = n \cdot \Delta \sim n \gg \sqrt{n}$.
- *Conclusion:* using a single distribution can be very misleading, especially if we want guaranteed results.
- Examples: high-risk application areas such as space exploration or nuclear engineering.

## 6. 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 $0 \notin \mathbf{x}_1$).

# 7. 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 $\mathbf{Y} \supseteq \mathbf{y}$.

# 8. 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.

## 9. 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 $\underline{r}_i \geq 0$, 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 $\underline{r} \geq 0$, 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 $2 \geq 0$.
- *Result:* $f([1, 2]) = [f(1), f(2)] = [-3, 0]$.
- *Comparison:* this is the exact range.

# 10. 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:* $f(\mathbf{x}) = [0, 0.25]$.

# 11. 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).

# 12. 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 $[0, 1]$.

## 13. 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.

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

## 15. 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:* $c_0 = a_0 \cdot b_0$, $c_i = a_0 \cdot b_i + b_0 \cdot a_i$,

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

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

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

# 16. 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 $x = 0.5 - \Delta x + [0, 0]$;
  - $\mathbf{r}_1 := 1 - (0.5 - \Delta) = 0.5 + \Delta x$;
  - $\mathbf{r}_2 := (0.5 - \Delta x) \cdot (0.5 + \Delta x)$, 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.

## 17. 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 $\Delta x_i$;
  - *idea:* keep, e.g., quadratic terms

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

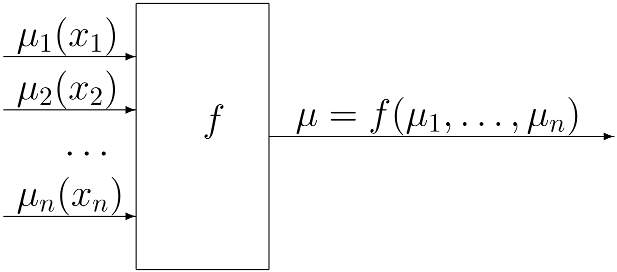
# 18. Interval Computations vs. Affine Arithmetic: Comparative Analysis

- *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.

## 19. Fuzzy Computations: A Problem




- *Given:* an algorithm $y = f(x_1, \dots, x_n)$ and $n$ fuzzy numbers $\mu_i(x_i)$.
- *Compute:* $\mu(y) = \max\limits_{x_1,\dots,x_n : f(x_1,\dots,x_n)=y} \min(\mu_1(x_1), \dots, \mu_n(x_n))$.
- *Motivation:* $y$ is a possible value of $Y \leftrightarrow \exists x_1, \dots, x_n$ s.t. each $x_i$ is a possible value of $X_i$ and $f(x_1, \dots, x_n) = y$.
- *Details:* "and" is min, $\exists$ ("or") is max, hence

$$\mu(y) = \max_{x_1,\dots,x_n} \min(\mu_1(x_1), \dots, \mu_n(x_n), t(f(x_1, \dots, x_n) = y)),$$

where $t(\text{true}) = 1$ and $t(\text{false}) = 0$.

## 20. Fuzzy Computations: Reduction to Interval Computations

- *Problem (reminder):*
  - *Given:* an algorithm $y = f(x_1, \ldots, x_n)$ and $n$ fuzzy numbers $X_i$ described by membership functions $\mu_i(x_i)$.
  - *Compute:* $Y = f(X_1, \ldots, X_n)$, where $Y$ is defined by Zadeh's extension principle:

$$\mu(y) = \max_{x_1,\ldots,x_n : f(x_1,\ldots,x_n)=y} \min(\mu_1(x_1), \ldots, \mu_n(x_n)).$$

- *Idea:* represent each $X_i$ by its $\alpha$-cuts

$$X_i(\alpha) = \{x_i : \mu_i(x_i) \geq \alpha\}.$$

- *Advantage:* for continuous $f$, for every $\alpha$, we have

$$Y(\alpha) = f(X_1(\alpha), \ldots, X_n(\alpha)).$$

- *Resulting algorithm:* for $\alpha = 0, 0.1, 0.2, \ldots, 1$ apply interval computations techniques to compute $Y(\alpha)$.

# 21. Case Study: Chip Design

- *Chip design:* one of the main objectives is to decrease the clock cycle.
- *Current approach:* uses worst-case (interval) techniques.
- *Problem:* the probability of the worst-case values is usually very small.
- *Result:* estimates are over-conservative – unnecessary over-design and under-performance of circuits.
- *Difficulty:* we only have *partial* information about the corresponding probability distributions.
- *Objective:* produce estimates valid for all distributions which are consistent with this information.
- *What we do:* provide such estimates for the clock time.

## 22. Estimating Clock Cycle: a Practical Problem

- *Objective:* estimate the clock cycle on the design stage.
- The clock cycle of a chip is constrained by the maximum path delay over all the circuit paths

$$D \stackrel{\text{def}}{=} \max(D_1, \ldots, D_N).$$

- The path delay $D_i$ along the $i$-th path is the sum of the delays corresponding to the gates and wires along this path.
- Each of these delays, in turn, depends on several factors such as:
  - the variation caused by the current design practices,
  - environmental design characteristics (e.g., variations in temperature and in supply voltage), etc.

# 23. Traditional (Interval) Approach to Estimating the Clock Cycle

- *Traditional approach:* assume that each factor takes the worst possible value.
- *Result:* time delay when all the factors are at their worst.
- *Problem:*
  - different factors are usually independent;
  - combination of worst cases is improbable.
- *Computational result:* current estimates are 30% above the observed clock time.
- *Practical result:* the clock time is set too high – chips are over-designed and under-performing.

## 24. Robust Statistical Methods Are Needed

- *Ideal case:* we know probability distributions.
- *Solution:* Monte-Carlo simulations.
- *In practice:* we only have *partial* information about the distributions of some of the parameters; usually:
  - the mean, and
  - some characteristic of the deviation from the mean – e.g., the interval that is guaranteed to contain possible values of this parameter.
- *Possible approach:* Monte-Carlo with several possible distributions.
- *Problem:* no guarantee that the result is a valid bound for all possible distributions.
- *Objective:* provide *robust* bounds, i.e., bounds that work for all possible distributions.

## 25. Towards a Mathematical Formulation of the Problem

- *General case:* each gate delay $d$ depends on the difference $x_1, \ldots, x_n$ between the actual and the nominal values of the parameters.
- *Main assumption:* these differences are usually small.
- Each path delay $D_i$ is the sum of gate delays.
- *Conclusion:* $D_i$ is a linear function: $D_i = a_i + \sum\limits_{j=1}^{n} a_{ij} \cdot x_j$ for some $a_i$ and $a_{ij}$.
- The desired maximum delay $D = \max\limits_i D_i$ has the form

$$D = F(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \max_i \left( a_i + \sum_{j=1}^{n} a_{ij} \cdot x_j \right).$$

## 26. Towards a Mathematical Formulation of the Problem (cont-d)

- *Known:* maxima of linear function are exactly convex functions:

$$F(\alpha \cdot x + (1-\alpha) \cdot y) \le \alpha \cdot F(x) + (1-\alpha) \cdot F(y)$$

  for all $x, y$ and for all $\alpha \in [0, 1]$;

- *We know:* factors $x_i$ are independent;
  - we know distribution of some of the factors;
  - for others, we know ranges $[\underline{x}_j, \overline{x}_j]$ and means $E_j$.
- *Given:* a convex function $F \ge 0$ and a number $\varepsilon > 0$.
- *Objective:* find the smallest $y_0$ s.t. for all possible distributions, we have $y \le y_0$ with the probability $\ge 1-\varepsilon$.

# 27. Additional Property: Dependency is Non-Degenerate

- *Fact:* sometimes, we learn additional information about one of the factors $x_j$.
- *Example:* we learn that $x_j$ actually belongs to a proper subinterval of the original interval $[\underline{x}_j, \overline{x}_j]$.
- *Consequence:* the class $\mathcal{P}$ of possible distributions is replaced with $\mathcal{P}' \subset \mathcal{P}$.
- *Result:* the new value $y_0'$ can only decrease: $y_0' \le y_0$.
- *Fact:* if $x_j$ is irrelevant for $y$, then $y_0' = y_0$.
- *Assumption:* irrelevant variables been weeded out.
- *Formalization:* if we narrow down one of the intervals $[\underline{x}_j, \overline{x}_j]$, the resulting value $y_0$ decreases: $y_0' < y_0$.

## 28. Formulation of the Problem

GIVEN:
- $n$, $k \le n$, $\varepsilon > 0$;
- a convex function $y = F(x_1, \ldots, x_n) \ge 0$;
- $n - k$ cdfs $F_j(x)$, $k + 1 \le j \le n$;
- intervals $\mathbf{x}_1, \ldots, \mathbf{x}_k$, values $E_1, \ldots, E_k$,

TAKE: all joint probability distributions on $R^n$ for which:

- all $x_i$ are independent,
- $x_j \in \mathbf{x}_j$, $E[x_j] = E_j$ for $j \le k$, and
- $x_j$ have distribution $F_j(x)$ for $j > k$.

FIND: the smallest $y_0$ s.t. for all such distributions, $F(x_1, \ldots, x_n) \le y_0$ with probability $\ge 1 - \varepsilon$.

WHEN: the problem is *non-degenerate* – if we narrow down one of the intervals $\mathbf{x}_j$, $y_0$ decreases.

## 29. Main Result and How We Can Use It

- *Result:* $y_0$ is attained when for each $j$ from 1 to $k$,
  - $x_j = \underline{x}_j$ with probability $\underline{p}_j \stackrel{\text{def}}{=} \dfrac{\overline{x}_j - E_j}{\overline{x}_j - \underline{x}_j}$, and
  - $x_j = \overline{x}_j$ with probability $\overline{p}_j \stackrel{\text{def}}{=} \dfrac{E_j - \underline{x}_j}{\overline{x}_j - \underline{x}_j}$.
- *Algorithm:*
  - simulate these distributions for $x_j$, $j < k$;
  - simulate known distributions for $j > k$;
  - use the simulated values $x_j^{(s)}$ to find

$$y^{(s)} = F(x_1^{(s)}, \ldots, x_n^{(s)});$$

  - sort $N$ values $y^{(s)}$: $y_{(1)} \le y_{(2)} \le \ldots \le y_{(N_i)}$;
  - take $y_{(N_i \cdot (1-\varepsilon))}$ as $y_0$.

## 30. Comment about Monte-Carlo Techniques

- *Traditional belief:* Monte-Carlo methods are inferior to analytical:
  - they are approximate;
  - they require large computation time;
  - simulations for *several* distributions, may mis-calculate the (desired) maximum over *all* distributions.
- *We proved:* the value corresponding to the selected distributions indeed provide the desired maximum value $y_0$.
- *General comment:*
  - justified Monte-Carlo methods often lead to *faster* computations than analytical techniques;
  - example: multi-D integration – where Monte-Carlo methods were originally invented.

## 31. Comment about Non-Linear Terms

- *Reminder:* in the above formula $D_i = a_i + \sum_{j=1}^{n} a_{ij} \cdot x_j$, we ignored quadratic and higher order terms in the dependence of each path time $D_i$ on parameters $x_j$.
- *In reality:* we may need to take into account some quadratic terms.
- *Idea behind possible solution:* it is known that the max $D = \max_i D_i$ of convex functions $D_i$ is convex.
- *Condition when this idea works:* when each dependence $D_i(x_1, \ldots, x_k, \ldots)$ is still convex.
- *Solution:* in this case,
  - the function function $D$ is still convex,
  - hence, our algorithm will work.

# 32. Case Study: Conclusions

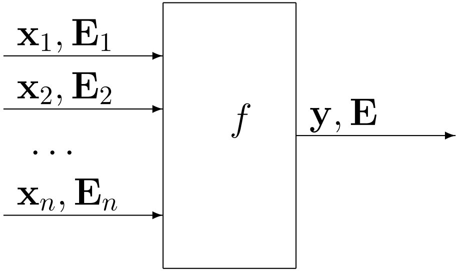
- *Problem of chip design:* decrease the clock cycle.
- *How this problem is solved now:* by using worst-case (interval) techniques.
- *Limitations of this solution:* the probability of the worst-case values is usually very small.
- *Consequence:* estimates are over-conservative, hence over-design and under-performance of circuits.
- *Objective:* find the clock time as $y_0$ s.t. for the actual delay $y$, we have $\text{Prob}(y > y_0) \le \varepsilon$ for given $\varepsilon > 0$.
- *Difficulty:* we only have *partial* information about the corresponding distributions.
- *What we have described:* a general technique that allows us, in particular, to compute $y_0$.

# 33. Combining Interval and Probabilistic Uncertainty: General Case

- *Problem:* there are many ways to represent a probability distribution.
- *Idea:* look for an objective.
- *Objective:* make decisions $E_x[u(x,a)] \to \max a$.
- *Case 1:* smooth $u(x)$.
- *Analysis:* we have $u(x) = u(x_0) + (x - x_0) \cdot u'(x_0) + \ldots$
- *Conclusion:* we must know moments to estimate $E[u]$.
- *Case of uncertainty:* interval bounds on moments.
- *Case 2:* threshold-type $u(x)$.
- *Conclusion:* we need cdf $F(x) = \text{Prob}(\xi \le x)$.
- *Case of uncertainty:* p-box $[\underline{F}(x), \overline{F}(x)]$.

## 34. Extension of Interval Arithmetic to Probabilistic Case: Successes

- *General solution:* parse to elementary operations $+$, $-$, $\cdot$, $1/x$, max, min.
- Explicit formulas for arithmetic operations known for intervals, for p-boxes $\mathbf{F}(x) = [\underline{F}(x), \overline{F}(x)]$, for intervals + 1st moments $E_i \stackrel{\text{def}}{=} E[x_i]$:



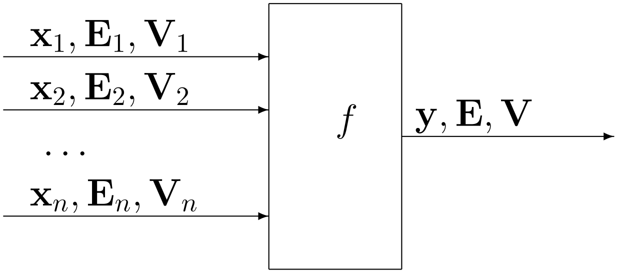
# 35. Successes (cont-d)

- *Easy cases:* $+$, $-$, product of independent $x_i$.
- *Example of a non-trivial case:* multiplication $y = x_1 \cdot x_2$, when we have no information about the correlation:
  - $\underline{E} = \max(p_1+p_2-1,0)\cdot\overline{x}_1\cdot\overline{x}_2+\min(p_1,1-p_2)\cdot\overline{x}_1\cdot\underline{x}_2+\min(1-p_1,p_2)\cdot\underline{x}_1\cdot\overline{x}_2+\max(1-p_1-p_2,0)\cdot\underline{x}_1\cdot\underline{x}_2$;
  - $\overline{E} = \min(p_1,p_2)\cdot\overline{x}_1\cdot\overline{x}_2+\max(p_1-p_2,0)\cdot\overline{x}_1\cdot\underline{x}_2+\max(p_2-p_1,0)\cdot\underline{x}_1\cdot\overline{x}_2+\min(1-p_1,1-p_2)\cdot\underline{x}_1\cdot\underline{x}_2$,

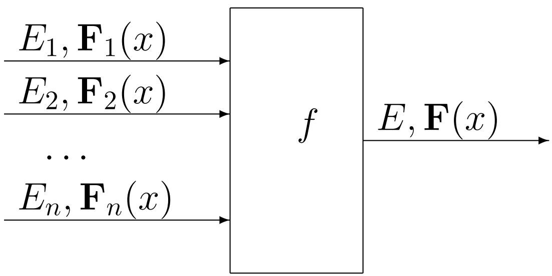
  where $p_i \stackrel{\text{def}}{=} (E_i - \underline{x}_i)/(\overline{x}_i - \underline{x}_i)$.

# 36. Challenges

- intervals + 2nd moments:




- moments + p-boxes; e.g.:



## 37. Case Study: Bioinformatics

- *Practical problem:* find genetic difference between cancer cells and healthy cells.
- *Ideal case:* we directly measure concentration $c$ of the gene in cancer cells and $h$ in healthy cells.
- *In reality:* difficult to separate.
- *Solution:* we measure $y_i \approx x_i \cdot c + (1 - x_i) \cdot h$, where $x_i$ is the percentage of cancer cells in $i$-th sample.
- *Equivalent form:* $a \cdot x_i + h \approx y_i$, where $a \stackrel{\text{def}}{=} c - h$.

# 38. Case Study: Bioinformatics (cont-d)

- *If we know $x_i$ exactly:* Least Squares Method $\sum\limits_{i=1}^n (a\cdot x_i+h-y_i)^2\to \min\limits_{a,h}$, hence $a=\dfrac{C(x,y)}{V(x)}$ and $h=E(y)-a\cdot E(x)$, where $E(x)=\dfrac{1}{n}\cdot\sum\limits_{i=1}^n x_i$,

$$V(x)=\frac{1}{n-1}\cdot\sum_{i=1}^n (x_i-E(x))^2,$$

$$C(x,y)=\frac{1}{n-1}\cdot\sum_{i=1}^n (x_i-E(x))\cdot(y_i-E(y)).$$

- *Interval uncertainty:* experts manually count $x_i$, and only provide interval bounds $\mathbf{x}_i$, e.g., $x_i\in[0.7,0.8]$.
- *Problem:* find the range of $a$ and $h$ corresponding to all possible values $x_i\in[\underline{x}_i,\overline{x}_i]$.

## 39. General Problem

- *General problem:*
  - we know intervals $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1], \ldots, \mathbf{x}_n = [\underline{x}_n, \overline{x}_n]$,
  - compute the range of $E(x) = \dfrac{1}{n}\sum_{i=1}^{n} x_i$, population variance $V = \dfrac{1}{n}\sum_{i=1}^{n}(x_i - E(x))^2$, etc.
- *Difficulty:* NP-hard even for variance.
- *Known:*
  - efficient algorithms for $\underline{V}$,
  - efficient algorithms for $\overline{V}$ and $C(x, y)$ for reasonable situations.
- *Bioinformatics case:* find intervals for $C(x, y)$ and for $V(x)$ and divide.

## 40. Case Study: Detecting Outliers

- In many application areas, it is important to detect *outliers*, i.e., unusual, abnormal values.
- In *medicine*, unusual values may indicate disease.
- In *geophysics*, abnormal values may indicate a mineral deposit (or an erroneous measurement result).
- In *structural integrity* testing, abnormal values may indicate faults in a structure.
- *Traditional engineering approach:* a new measurement result $x$ is classified as an outlier if $x \notin [L, U]$, where

$$L \stackrel{\text{def}}{=} E - k_0 \cdot \sigma, \quad U \stackrel{\text{def}}{=} E + k_0 \cdot \sigma,$$

  and $k_0 > 1$ is pre-selected.
- *Comment:* most frequently, $k_0 = 2$, 3, or 6.

# 41. Outlier Detection Under Interval Uncertainty: A Problem

- In some practical situations, we only have intervals $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$.
- Different $x_i \in \mathbf{x}_i$ lead to different intervals $[L, U]$.
- A *possible* outlier: outside *some* $k_0$-sigma interval.
- *Example:* structural integrity – not to miss a fault.
- A *guaranteed* outlier: outside *all* $k_0$-sigma intervals.
- *Example:* before a surgery, we want to make sure that there is a micro-calcification.
- A value $x$ is a possible outlier if $x \notin [\overline{L}, \underline{U}]$.
- A value $x$ is a guaranteed outlier if $x \notin [\underline{L}, \overline{U}]$.
- *Conclusion:* to detect outliers, we must know the ranges of $L = E - k_0 \cdot \sigma$ and $U = E + k_0 \cdot \sigma$.

## 42. Outlier Detection Under Interval Uncertainty: A Solution

- *We need:* to detect outliers, we must compute the ranges of $L = E - k_0 \cdot \sigma$ and $U = E + k_0 \cdot \sigma$.
- *We know:* how to compute the ranges $\mathbf{E}$ and $[\underline{\sigma}, \overline{\sigma}]$ for $E$ and $\sigma$.
- *Possibility:* use interval computations to conclude that $L \in \mathbf{E} - k_0 \cdot [\underline{\sigma}, \overline{\sigma}]$ and $L \in \mathbf{E} + k_0 \cdot [\underline{\sigma}, \overline{\sigma}]$.
- *Problem:* the resulting intervals for $L$ and $U$ are *wider* than the actual ranges.
- *Reason:* $E$ and $\sigma$ use the same inputs $x_1, \ldots, x_n$ and are hence not independent from each other.
- *Practical consequence:* we miss some outliers.
- *Desirable:* compute *exact* ranges for $L$ and $U$.
- *Application:* detecting outliers in gravity measurements.

# 43. Computing Amount of Information: A Problem

- *Uncertainty*: usually, there are several ($n$) different states which are consistent with our knowledge.
- *Question:* how much information we need to gain to determine the actual state of the world?
- *Natural measure:* average number of "yes"-"no" questions that we need to ask.
- *Probabilistic case:* sometimes, we know the probabilities $p_1, \ldots, p_n$ of different states.
- *Shannon's result:* $S = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$.
- *Problem:* often, we only know intervals $\mathbf{p}_i = [\underline{p}_i, \overline{p}_i]$ of possible values of $p_i$.
- *Question:* find the range $\mathbf{S} = [\underline{S}, \overline{S}]$ of possible values of $S$.

## 44. Computing Amount of Information: Results

- *Problem (reminder):*
  - *given:* intervals $\mathbf{p}_i = [\underline{p}_i, \overline{p}_i]$ of possible values of $p_i$.
  - *find:* the range $\mathbf{S} = [\underline{S}, \overline{S}]$ of possible values of $S = -\sum\limits_{i=1}^{n} p_i \cdot \log_2(p_i)$.
- *Results:*
  - the problem of computing $\mathbf{S}$ is, in general, NP-hard;
  - algorithms that efficiently compute $\mathbf{S}$ in many practically important situations.

# 45. Acknowledgments

This work was supported in part by:

- NASA under cooperative agreement NCC5-209,
- NSF grants EAR-0225670 and DMS-0532645,
- Star Award from the University of Texas System, and
- Texas Department of Transportation grant No. 0-5453.

# 46. Proof of the Chip Result

- Let us fix the optimal distributions for $x_2, \ldots, x_n$; then,

$$\text{Prob}(D \le y_0) = \sum_{(x_1,\ldots,x_n): D(x_1,\ldots,x_n) \le y_0} p_1(x_1) \cdot p_2(x_2) \cdot \ldots$$

- So, $\text{Prob}(D \le y_0) = \sum\limits_{i=0}^{N} c_i \cdot q_i$, where $q_i \stackrel{\text{def}}{=} p_1(v_i)$.

- Restrictions: $q_i \ge 0$, $\sum\limits_{i=0}^{N} q_i = 1$, and $\sum\limits_{i=0}^{N} q_i \cdot v_i = E_1$.

- Thus, the worst-case distribution for $x_1$ is a solution to the following linear programming (LP) problem:

Minimize $\sum\limits_{i=0}^{N} c_i \cdot q_i$ under the constraints $\sum\limits_{i=0}^{N} q_i = 1$ and

$$\sum_{i=0}^{N} q_i \cdot v_i = E_1,\ q_i \ge 0, \quad i = 0, 1, 2, \ldots, N.$$

## 47. Proof of the Chip Result (cont-d)

- *Minimize:* $\sum\limits_{i=0}^{N} c_i \cdot q_i$ under the constraints $\sum\limits_{i=0}^{N} q_i = 1$ and $\sum\limits_{i=0}^{N} q_i \cdot v_i = E_1$, $q_i \geq 0$, $\;i = 0, 1, 2, \ldots, N$.
- *Known:* in LP with $N+1$ unknowns $q_0, q_1, \ldots, q_N$, $\geq N+1$ constraints are equalities.
- *In our case:* we have 2 equalities, so at least $N-1$ constraints $q_i \geq 0$ are equalities.
- Hence, no more than 2 values $q_i = p_1(v_i)$ are non-0.
- If corresponding $v$ or $v'$ are in $(\underline{x}_1, \overline{x}_1)$, then for $[v, v'] \subset \mathbf{x}_1$ we get the same $y_0$ – in contradiction to non-degeneracy.
- Thus, the worst-case distribution is located at $\underline{x}_1$ and $\overline{x}_1$.
- The condition that the mean of $x_1$ is $E_1$ leads to the desired formulas for $\underline{p}_1$ and $\overline{p}_1$.