Processing Interval Sensor Data in the Presence of Outliers, with Potential Applications to Localizing Underwater Robots

Jan Sliwka and Luc Jaulin
Bureau D214
ENSTA-Bretagne, Brest, France
{luc.jaulin,jan.sliwka}@ensta-bretagne.fr

Martine Ceberio and Vladik Kreinovich
Department of Computer Science
University of Texas, El Paso, Texas, USA
{mceberio,vladik}@utep.edu

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

1. Need to Consider Interval Uncertainty

- The value $\widetilde{x}$ measured by a sensor is, in general, different from the actual (unknown) value x.
- Traditionally, in science and engineering, it is assumed that we know the probability distribution of

$$\Delta x \stackrel{\text{def}}{=} \widetilde{x} - x.$$

- However, in many real-life situations, we only know the upper bound Δ on the measurement error: $|\Delta x| \le \Delta$.
- In this case, the only information that we have about the actual value x is that $x \in \mathbf{x} = [\underline{x}, \overline{x}]$, where

$$\underline{x} = \widetilde{x} - \Delta \text{ and } \overline{x} = \widetilde{x} + \Delta.$$

- It is therefore important to consider interval uncertainty.

2. Need to Consider Outliers

- These exist many efficient techniques for processing such interval data.
- These techniques form an important part of *granular computing*.
- In practice, sensor malfunction sometimes produces *outliers*, values outside the interval $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$.
- Outliers are usually characterized by a *proportion* ε of measurement results that could be erroneous.
- For example, $\varepsilon = 0.1$ means that at least $\alpha = 1 - \varepsilon = 90\%$ of the intervals contain the actual values.
- Sometimes, we do not know ε, so we should produce results corresponding to different values ε.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

3. Combining Interval Uncertainty and Outliers: What is Known

- In general, outliers turn easy-to-solve interval problems into difficult-to-solve (NP-hard) ones.
- This is true even in the simplest case, when we simply repeatedly measure several quantities $x_1, \ldots, x_d$.
- After each measurement, we get a box
$$X^{(j)} = [\underline{x}_1^{(j)}, \overline{x}_1^{(j)}] \times \ldots \times [\underline{x}_d^{(j)}, \overline{x}_d^{(j)}].$$
- In the absence of outliers, the actual state belongs to the easy-to-compute intersection of all these boxes.
- With outliers, the problem becomes NP-hard :-(
- For fixed d, there is a polynomial-time algorithm for solving this problem; its running time is $O(n^d)$.
- However, this running time grows exponentially with the dimension d.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

4. Case Study: Localizing Underwater Robots

- For underwater robot localization, we use distance measurements produced by sonars.
- A sonar measures echoes from the desired object *and* from other objects along the path.
- *Example:* a robot tries to localize itself by measuring the distance to the nearest wall.
- *Problem:* the sensor may detect a reflection from the surface wave, a fish or a diver – producing an outlier.
- After several measurements, we get a significant number of outliers.

Need to Consider ...
Need to Consider Outliers
Combining Interval ...
Case Study: ...
A Natural Fuzzy ...
Joint Processing of ...
Joint Processing of ...
How to Compute the ...
Fast Algorithm for ...

5. Efficient Algorithm for the Simplest Situation

- We have n interval measurements $[\underline{x}^{(j)}, \overline{x}^{(j)}]$ of the same quantity x.
- We know the upper bound $\varepsilon > 0$ on the proportion of measurements which are outliers.
- This means that the actual (unknown) value x satisfies at least $n \cdot (1-\varepsilon)$ of n constraints $\underline{x}^{(j)} \le x \le \overline{x}^{(j)}$.
- To find the set of all such x, we sort all the endpoints $\underline{x}^{(j)}$ and $\overline{x}^{(j)}$ into a sequence $x_{(1)} \le x_{(2)} \le \ldots \le x_{(2n)}$.
- This divides the set of possible values of x into $2n-1$ zones $[x_{(1)}, x_{(2)}]$, $[x_{(2)}, x_{(3)}]$, ..., $[x_{(2n-1)}, x_{(2n)}]$.
- For each zone k, we count the number of constraints c_k which are satisfied for elements of this zone.
- If $c_k \ge n \cdot (1-\varepsilon)$, we add k-th zone to the desired set.
- This takes time $O(n \cdot \log(n)) + O(n) = O(n \cdot \log(n))$.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

6. What if we Are Only Interested in the Interval of Possible Values

- *Example:*
  - as a result of measuring the same quantity, we get two intervals $[-2,-1]$ and $[1,2]$;
  - since their intersection is empty, we know that one of them is an outlier;
  - let us assume that we know that one of them is correct.
- In this case, the set of all possible values of x is the set $[-2,-1] \cup [1,2]$.
- In this situation, the smallest possible value of x is -2, and the largest possible value of x is 2.
- Thus, the smallest interval that contains all possible values of x is the interval $[-2,2]$.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

7. Computing Interval of Possible Values: Analysis of the Problem and the Resulting Algorithm

- Let us sort all n upper endpoints $\overline{x}^{(j)}$, $1 \le j \le n$, into an increasing sequence $u_1 \le u_2 \le \ldots \le u_n$.
- We can guarantee that x is smaller than or equal to at least $n \cdot (1 - \varepsilon)$ terms in this sequence; so, $x_i \le u_{n \cdot \varepsilon}$.
- Similarly, if we sort $\underline{x}^{(j)}$, $1 \le j \le n$, into $\ell_1 \le \ell_2 \le \ldots \le \ell_n$, then we conclude that $x \ge \ell_{n \cdot (1-\varepsilon)}$.
- Thus, we can conclude that the desired interval of possible values x is equal to $[\ell_{n \cdot (1-\varepsilon)}, u_{n \cdot \varepsilon}]$.
- Finding elements of a given rank can be done in linear time $O(n)$.
- Our preliminary experiments confirm that this technique correctly locates the underwater robot.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

8. A Natural Fuzzy Representation of Our Problem

- For each x, we can find the proportion $\mu(x) \in [0, 1]$ of constraints which are satisfied for this x.
- It is reasonable to interpret the resulting function $\mu(x)$ as a membership function.
- The actual value x must belong to the set of all the values x for which $\mu(x) \geq 1 - \varepsilon$.
- Thus, the set of all possible x is the α-cut, with

$$\alpha = 1 - \varepsilon.$$

- Up to now, fuzzy sets are just an interpretation.
- We will see that by using known fuzzy algorithms, we can speed up computations.
- Thus, fuzzy interpretation is indeed helpful.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

9. Outliers Beyond Counting

- In real life, we may have more confidence in some constraints, and less confidence in other constraints.
- Let p_i be a probability that the i-th constraint is satisfied; we assume that constraints are independent.
- Let X_i be the set of all the values that satisfy the i-th constraint.
- Then, for each x, the probability that x is a possible value is $p(x) = \prod\limits_{i:x\in X_i} p_i \cdot \prod\limits_{j:x\notin X_j} (1-p_j)$.
- As usual in prob. approaches, we decide that only states x with $p(x) \geq p_0$ are possible, for some threshold p_0.
- $p(x) \geq p_0 \Leftrightarrow \sum\limits_{i:x\in X_i} w_i \geq t_0$, with $w_i = \ln(p_i/(1-p_i))$.
- In fuzzy terms, this is equivalent to taking $\mu(x)$ as the total weight of all the constraints satisfied by x.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

10. Data Processing under Outliers is, in General, NP-Hard

- *Example:* it is not possible to directly measure the 3D spatial coordinates y_j of an underwater robot.
- However, we can reconstruct y_j if we measure the distances x_i from the robot to several known objects.
- *In general:* we need to process the measurement results.
- Under constraints, the problem is NP-hard even for linear data processing:
  - given the values a_{ij}, x_i, and $\varepsilon \in (0, 1)$,
  - check whether out of n constraints $\sum\limits_{j=1}^{d} a_{ij} \cdot y_j = x_i$, we can select a consistent set of $n \cdot (1 - \varepsilon)$ ones.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

11. Joint Processing of Several Quantities: Case when Sensors are of Different Type

- *General case:* we measure quantities $x_1, \ldots, x_d$, and we use a known relation $y = f(x_1, \ldots, x_d)$ to estimate y.
- *Situation:* measurements of different x_i are done by *different* types of sensors.
- *In this case:* for each sensor type, we have its own upper bound ε_i on the frequency of outliers.
- Based on the bound ε_i, we can compute the interval $[\underline{x}_i, \overline{x}_i]$ of possible values of x_i.
- We can then use interval computation techniques to estimate the range

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

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

12. Joint Processing of Several Quantities: Case when Sensors are of the Same Type

- Simplest case: $d = 2$, $f(x_1, x_2) = x_1 + x_2$.
- Let α_i is the proportion of x_i-sensors that work well.
- For each α_i, we have the lower bound $\underline{x}_i(\alpha_i)$ for x_i.
- For these α_i, the sum $y = x_1 + x_2$ is bounded from below by the sum $\underline{x}_1(\alpha_1) + \underline{x}_2(\alpha_2)$.
- We do not know α_i, we only know that

$$\alpha_1 \cdot \frac{n_1}{n_1 + n_2} + \alpha_2 \cdot \frac{n_2}{n_1 + n_2} = \alpha.$$

- Thus, we can conclude that y is larger than one of such sums – hence larger than the smallest of these sums:

$$\underline{y}(\alpha) = \min \left\{ \underline{x}_1(\alpha_1) + \underline{x}_2(\alpha_2) : \alpha_1 \cdot \frac{n_1}{n_1 + n_2} + \alpha_2 \cdot \frac{n_2}{n_1 + n_2} = \alpha \right\}$$

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

13. How to Compute the Bounds $\underline{y}(\alpha)$ and $\overline{y}(\alpha)$

- The upper bound for $y = x_1 + x_2$ is minus the lower bound for $-y = (-x_1) + (-x_2)$.
- Thus, computing $\overline{y}$ can be reduced to computing $\underline{y}$.
- The formula for $\underline{y}(\alpha)$ can be simplified if we take
$$t_i \stackrel{\text{def}}{=} \alpha_i \cdot \frac{n_i}{n_1 + n_2} \text{ and } f_i(t_i) \stackrel{\text{def}}{=} \underline{x}_i\left(t_i \cdot \frac{n_1 + n_2}{n_i}\right):$$
$$\underline{y}(\alpha) = \min_{t_1, t_2 : t_1 + t_2 = \alpha} (f_1(t_1) + f_2(t_2)),$$
- This formula is similar to Zadeh's extension principle for $f_{\&}(a, b) = a \cdot b$: $\mu(t) = \max\limits_{t_1, t_2:\ t_1 + t_2 = t} (\mu_1(t_1) \cdot \mu_2(t_2))$.
- *Difference:* we need addition and min, Zadeh's formula uses multiplication and max.
- *Idea:* use $\exp(-x)$: take $\mu_i(t_i) = \exp(-f_i(t_i))$, find $\mu(t)$, then compute $\underline{y}(t) = -\ln(\mu(t))$.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

14. Straightforward Computation of $\mu(t)$

- How to compute $\mu(t) = \max\limits_{t_1,t_2:\ t_1+t_2=t} (\mu_1(t_1) \cdot \mu_2(t_2))$?
- In reality, we only know finitely many (n) values of $\mu_1(x)$ and $\mu_2(x)$.
- In this case, it is reasonable to compute only n values of $\mu(t)$.
- For each of these n values, according to the formula, we must find the largest of n products.
- Computing each product takes one step.
- So, computing one value of $\mu(t)$ takes $O(n)$ steps.
- Thus, to compute *all* n values of function $\mu(t)$, we need $n \cdot O(n) = O(n^2)$ steps.
- For large n, this number is large, so faster methods are needed.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

15. A Faster Algorithm for Computing $\mu(t)$: Idea

- We want to compute $\mu(t) = \max\limits_{t_1,t_2:\ t_1+t_2=t} (\mu_1(t_1)\cdot\mu_2(t_2))$.
- Known fact: for $\mu_i \ge 0$, we have
$$\max(\mu_1,\ldots,\mu_n) = \lim_{p\to\infty} (\mu_1^p + \ldots + \mu_n^p)^{1/p}.$$
- So, for sufficiently large p, we have
$$\max(\mu_1,\ldots,\mu_n) \approx (\mu_1^p + \ldots + \mu_n^p)^{1/p}.$$
- So, $\mu(t) \approx M(t)^{1/p}$, w/ $M(t) \stackrel{\rm def}{=} \sum\limits_{t_1} \mu_1(t_1))^p\cdot(\mu_2(t-t_1))^p$.
- When t_1 are equally spaced, $M(t)$ a *convolution* of $M_1(x) = (\mu_1(x))^p$ and $M_2(x) = (\mu_2(x))^p$.
- *Known fact:* Fourier transform of the convolution $M_1 * M_2$ is the product of the Fourier transforms.
- *Known fact:* Fourier transform can be computed in time $O(n\log(n))$ (*Fast Fourier Transform*).

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

16. Resulting Algorithm

- First, we pick a large number p.
- For each of n values t_1, we compute the values

$$M_1(x) = (\mu_1(x))^p \text{ and } M_2(x) = (\mu_2(x))^p.$$

- We apply FFT to the functions $M_1(x)$ and $M_2(x)$ and get $\widehat{M_1}(\omega)$ and $\widehat{M_2}(\omega)$ (for n different values ω).
- We multiply $\widehat{M_1}(\omega)$ and $\widehat{M_2}(\omega)$; let us denote the corresponding product by $\widehat{M}(\omega)$.
- We apply inverse Fast Fourier transform to the product $\widehat{M}(\omega)$, and get $M(t)$.
- Finally, we reconstruct $\mu(t)$ as $(M(t))^{1/p}$.
- FFT takes time $O(n \cdot \log(n))$, all other steps are $O(n)$, so overall, we need $O(n \cdot \log(n)) \ll n^2$ steps.

Need to Consider ...
Need to Consider Outliers
Combining Interval ...
Case Study: ...
A Natural Fuzzy ...
Joint Processing of ...
Joint Processing of ...
How to Compute the ...
Fast Algorithm for ...

17. General Case: Analysis of the Problem

- For every i, pick some "mean" value $\widetilde{x}_i$.
- Then, $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i \in [\Delta_i^-(\alpha_i), \Delta_i^+(\alpha_i)]$, where

$$\Delta_i^-(\alpha_i) \stackrel{\text{def}}{=} \widetilde{x}_i - \overline{x}_i(\alpha_i) \text{ and } \Delta_i^+(\alpha_i) \stackrel{\text{def}}{=} \widetilde{x}_i - \overline{x}_i(\alpha_i).$$

- Measurements are reasonably accurate.
- Hence, for estimating $\Delta y = \widetilde{y} - y$, we can only keep terms linear in Δx_i.
- So, $\Delta y = f(\widetilde{x}_1, \ldots, \widetilde{x}_d) - f(x_1, \ldots, x_d) \approx \sum\limits_{i=1}^{d} c_i \cdot \Delta x_i$, where $c_i \stackrel{\text{def}}{=} \dfrac{\partial f}{\partial x_i}(\widetilde{x}_1, \ldots, \widetilde{x}_d)$.
- Thus, the smallest possible value of y is equal to $\widetilde{y} + \Delta^-$, where $\Delta^- = \sum\limits_{i=1}^{d} |c_i| \cdot (-\Delta_i^{s_i}(\alpha_i))$ and $s_i \stackrel{\text{def}}{=} \operatorname{sign}(c_i)$.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

18. Analysis of the Problem (cont-d)

- *Reminder:* $\Delta^- = \sum\limits_{i=1}^{d} |c_i| \cdot (-\Delta_i^{s_i}(\alpha_i))$.
- *In general:* $t_1 + \ldots + t_d = \alpha$, where $t_i \stackrel{\text{def}}{=} \alpha_i \cdot \dfrac{n_i}{n}$.
- Thus, $\Delta^- = \sum\limits_{i=1}^{d} f_i(t_i)$, where $f_i(t_i) \stackrel{\text{def}}{=} -|c_i| \cdot \Delta_i^{s_i}\left(t_i \cdot \dfrac{n}{n_i}\right)$.
- We do not know the values α_i, we only know that there are *some* values that satisfy the above equation.
- Thus, the smallest possible value y is attained when Δ^- takes the smallest possible value:

$$\Delta^-(\alpha) = \min_{t_1,\ldots,t_d:\ t_1+\ldots+t_d=\alpha} \sum_{i=1}^{d} f_i(t_i).$$

Need to Consider ...
Need to Consider Outliers
Combining Interval ...
Case Study: ...
A Natural Fuzzy ...
Joint Processing of ...
Joint Processing of ...
How to Compute the ...
Fast Algorithm for ...

19. Reduction to Fuzzy Computations: Idea

- *Reminder:* $\Delta^-(\alpha) = \min\limits_{t_1,\ldots,t_d:\ t_1+\ldots+t_d=\alpha} \sum\limits_{i=1}^{d} f_i(t_i)$.
- *We want:* to reduce this formula to Zadeh's extension principle $\mu(t) = \max\limits_{t_1,\ldots,t_d:\ t_1+\ldots+t_d=t} \prod\limits_{i=1}^{d} \mu_i(t_i)$.
- *Idea:* take $\mu(t) = \exp(-\Delta^-(t))$ and $\mu_i(t_i) = \exp(-f_i(t_i))$.
- *Resulting algorithm:*
  - for each i, we select $\widetilde{x}_i$ and compute $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_d)$, $c_i = \dfrac{\partial f}{\partial x_i}(\widetilde{x}_1, \ldots, \widetilde{x}_d)$ and $s_i = \text{sign}(c_i)$;
  - compute $f_i(t_i) \stackrel{\text{def}}{=} -|c_i| \cdot \Delta_i^{s_i}\left(t_i \cdot \dfrac{n}{n_i}\right)$ and $\mu_i(t_i) = \exp(-f_i(t_i))$;
  - apply a fuzzy algorithm $\mu_1(t_1), \ldots, \mu_d(t_d) \to \mu(t)$;
  - compute $\Delta^-(t) = -\ln(\mu(t))$ and $\underline{y}(\alpha) = \widetilde{y} + \Delta^-(\alpha)$.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

20. Fast Algorithm for Computing $\mu(t)$

- We want to compute $\mu(t) = \max\limits_{t_1,\ldots,t_d:\ t_1+\ldots+t_d=t} \prod\limits_{i=1}^{d} \mu_i(t_i)$.
- We pick a large number p, and compute the values $M_i(x) = (\mu_i(x))^p$ for all i and all x.
- We apply FFT to the functions $M_i(x)$ and get $\widehat{M_i}(\omega)$ (for n different values ω).
- We multiply the functions $\widehat{M_i}(\omega)$; let us denote the corresponding product by $\widehat{M}(\omega)$.
- We apply inverse Fast Fourier transform to the product $\widehat{M}(\omega)$, and get $M(t)$.
- Finally, we reconstruct $\mu(t)$ as $(M(t))^{1/p}$.
- FFT takes time $O(n \cdot \log(n))$, all other steps are $O(n)$, so overall, we need $O(n \cdot \log(n))$ steps.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

21. Acknowledgment

This work was partly supported

- by a MRIS (Office of Advanced Research and Innovation) grant from the Diréction Generale de l'Armement (DGA), France,
- by the US National Science Foundation grants HRD-0734825 and DUE-0926721, and
- by Grant 1 T36 GM078000-01 from the US National Institutes of Health.
- Jan Sliwka was also supported by Brest Métropole Océane (BMO).

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

22. Appendix: Proof of NP-hardness

- A standard way to prove an NP-hardness of a problem is to reduce one of the known NP-hard problems to it.
- As such a known NP-hard problem, we take the *subset sum* problem:
  - given positive integers $s_1, \ldots, s_m$, and s,
  - check whether $s = \sum\limits_{i=1}^{m} \varepsilon_i \cdot s_i = s$ for some $\varepsilon_i \in \{0, 1\}$.
- We will reduce each instance of this problem to the following problem, with $n = m/\varepsilon$ constraints:
  - $2m$ constraints $y_1 = 0$, $y_1 = 1$, $\ldots$, $y_m = 0$, $y_m = 1$;
  - $n - 2m$ identical constraints $\sum s_i \cdot y_i = s$.
- Since $0 \neq 1$, out of each pair of constraints $y_i = 1$ and $y_i = 1$, only one can be satisfied.
- So, at most $n - m$ constraints can be satisfied.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .

23. Proof of NP-hardness (cont-d)

- If the subset sum problem has a solution, then:
  - all $n - 2m$ constraints $\sum s_i \cdot y_i = s$ are satisfied;
  - for each i, either $y_i = 0$ constraint or $y_i = 1$ constraint is satisfied,
- So, $n - m = n \cdot (1 - \varepsilon)$ constraints are satisfied.
- Vice versa, if $n - m$ constraints are satisfied, then at most m constraints must be violated.
- Thus, for every i, we must have $y_i = 0$ and $y_i = 1$ and we will also have $\sum s_i \cdot y_i = s$.
- So, we have a solution to the original subset sum problem.
- The reduction is proven, so our problem is indeed NP-hard.

Need to Consider . . .
Need to Consider Outliers
Combining Interval . . .
Case Study: . . .
A Natural Fuzzy . . .
Joint Processing of . . .
Joint Processing of . . .
How to Compute the . . .
Fast Algorithm for . . .