Fast Algorithms for Computing Statistics under Interval Uncertainty: An Overview

Vladik Kreinovich and Gang Xiang

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

1. Outline

- Formulation of the problem: computing statistics under interval uncertainty.
- Analysis of the problem.
- Reasonable classes of problems for which we can expect feasible algorithms for statistics of interval data.
- Overview of the classes.
- A sample result: linear algorithm for computing variance under interval uncertainty.
- Applications.

2. Computing Statistics is Important

- In many engineering applications, we are interested in computing statistics.
- *Example:* we observe a pollution level $x(t)$ in a lake at different moments of time t.
- *Objective:* estimate standard statistical characteristics: mean E, variance V, correlation w/other measurements.
- For each of these characteristics C, there is an estimate $C(x_1, \ldots, x_n)$ based on the observed values $x_1, \ldots, x_n$.
- *Sample average* $E(x_1, \ldots, x_n) = \dfrac{1}{n} \cdot \sum\limits_{i=1}^{n} x_i.$
- *Sample variance* $V(x_1, \ldots, x_n) = \dfrac{1}{n} \cdot \sum\limits_{i=1}^{n} (x_i - E)^2.$

3. Interval Uncertainty

- *Interval uncertainty in measurements:*
  - often, we only know the approximate (measured) value $\widetilde{x}_i$ and the measurement accuracy Δ_i;
  - the actual (unknown) value of x_i is in

$$\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i].$$

- *Interval uncertainty in observations:*
  - *example:* on the 5-th day, the seed did not germinate, on the 6-th day it germinated;
  - *conclusion:* $t \in [5, 6]$.
- *Intervals from need to protect privacy:*
  - instead of recording the exact values of salary, age, etc.,
  - we only store the range: e.g., age from 10 to 20, from 20 to 30, etc.

4. Estimating Statistics Under Interval Uncertainty: A Problem

- *Situation:* in many cases, we only know the intervals

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

- *Problem:* different values $x_i \in \mathbf{x}_i$ lead to different values of the statistical characteristic $C(x_1, \ldots, x_n)$.
- *Conclusion:* a reasonable estimate for the corresponding statistical characteristic is the range

$$C(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=} \{C(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

- *Task:* modify the existing statistical algorithms so that they compute these ranges.
- This is the problem that we will be handling in the talk.

5. Precise Formulation of the Problem: Estimating Statistics Under Interval Uncertainty

- *Given:*
  - n intervals $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1], \ldots, \mathbf{x}_n = [\underline{x}_n, \overline{x}_n]$;
  - a statistical characteristic $C(x_1, \ldots, x_n)$.
- *Comment:* each interval $\mathbf{x}_i$ contains the actual (unknown) value x_i of the quantity x_i.
- *Compute:* the range

$$C(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=} \{C(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$$

of possible values of $C(x_1, \ldots, x_n)$ when $x_i \in \mathbf{x}_i$.

6. Analysis of the Problem

- *Known fact:* for some characteristics, solving this problem is straightforward.
- *Example:* the sample mean $E = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$ is monotonic in each of n variables x_i.
- *Conclusion:* to find the range $[\underline{E}, \overline{E}] = E(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, we compute $\underline{E} = \frac{1}{n} \cdot \sum_{i=1}^{n} \underline{x}_i$ and $\overline{E} = \frac{1}{n} \cdot \sum_{i=1}^{n} \overline{x}_i$.
- *Known fact:* for some characteristics, solving this problem is difficult.
- *Example:* computing the range $[\underline{V}, \overline{V}] = V(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is, in general, NP-hard.

7. Linearization

- *General idea:* uncertainty comes from measurement errors $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$ (so that $x_i = \widetilde{x}_i - \Delta x_i$).
- *Frequent situation:* measurement errors are small.
- *Engineering approach:* expand $C(x_1, \ldots, x_n)$ in Taylor series at $\widetilde{x}_i \stackrel{\text{def}}{=} (\underline{x}_i + \overline{x}_i)/2$ and keep only linear terms:

$$C_{\text{lin}}(x_1, \ldots, x_n) = C_0 - \sum_{i=1}^{n} C_i \cdot \Delta x_i,$$

where $C_0 \stackrel{\text{def}}{=} C(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ and $C_i \stackrel{\text{def}}{=} \dfrac{\partial C}{\partial x_i}(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

- *Resulting estimate:* we estimate the range of C as $[C_0 - \Delta, C_0 + \Delta]$, where $\Delta \stackrel{\text{def}}{=} \sum\limits_{i=1}^{n} |C_i| \cdot \Delta_i$.
- *Shortcoming:* the intervals are sometimes wide, so that high order terms can no longer be ignored.

8. Straightforward Interval Computations

- *Main idea:* inside the computer, every algorithm consists of elementary operations $f(a,b)$.
- *Fact:* for each $f(a,b)$, once we know the intervals $\mathbf{a}$ and $\mathbf{b}$, we can compute the exact range $f(\mathbf{a},\mathbf{b})$.
- *Straightforward interval computations:* replacing each operation $f(a,b)$ by the corr. interval operation.
- *Known:* as a result, we get an enclosure for the desired range.
- *Problem:* we get excess width. Example:
  - For $\mathbf{x}_1 = \mathbf{x}_2 = [0,1]$, the actual $V = \dfrac{(x_1 - x_2)^2}{4}$ and hence, the actual range $\mathbf{V} = [0, 0.25]$.
  - On the other hand, $\mathbf{E} = [0,1]$, hence

$$\frac{(\mathbf{x}_1 - \mathbf{E})^2 + (\mathbf{x}_2 - \mathbf{E})^2}{2} = [0,1] \supset [0, 0.25].$$

9. For this Problem, Traditional Optimization Methods Sometimes Require Unreasonably Long Time

- *Typical problem:* compute the exact range $[\underline{V}, \overline{V}]$ of the finite sample variance.
- *Natural idea:* solve this problem as a constrained optimization problem.
- *Formulation:* $V \to \min$ (or $V \to \max$) under the constraints

$$\underline{x}_1 \le x_1 \le \overline{x}_1, \ldots, \underline{x}_n \le x_n \le \overline{x}_n.$$

- *Known:* optimization techniques can compute "sharp" (exact) values of $\min(f(x))$ and $\max(f(x))$.
- *Problem:* general constrained optimization algorithms can require exponential time.
- *Difficulty:* for $n \approx 300$, the value 2^n becomes larger than the lifetime of the Universe.

10. Analysis of the Problem: Summary

- *Problem (reminder):* compute the range $\mathbf{C}$ of a statistical characteristic C under interval uncertainty.
- *Deficiencies of the existing methods:* they are
  - either not always efficient,
  - or do not always provide us with sharp estimates.
- *Conclusion:* we need new methods.
- *Main part of our talk:*
  - *characteristic:* sample variance V;
  - *classes of problems:* all previously proposed practically important classes;
  - *what we do:* describe fast methods for computing $\mathbf{V}$ for all these classes.
- *Additional results:* we describe fast algorithms for several other statistical characteristics.

11. Practically Important Classes of Problems

1. *Narrow intervals:* intervals $\mathbf{x}_i$ do not intersect with each other.
2. *Slightly wider narrow intervals:* for some $K > 2$, each collection of K intervals $\mathbf{x}_i$ has an empty intersection.
3. *Single MI:* no $\mathbf{x}_i$ is a proper subinterval of the (interior of the) other, i.e., $[\underline{x}_i, \overline{x}_i] \not\subseteq (\underline{x}_j, \overline{x}_j)$.
4. *Several MI:* intervals $\mathbf{x}_i$ can be divided into m subgroups, with a single MI property for each subgroup.
5. *Privacy case:* we fix values $x_{(1)} < x_{(2)} < \ldots < x_{(m)}$, and allow only intervals $[x_{(k)}, x_{(k+1)}]$.
6. *Non-detects:* each non-degenerate $[\underline{x}_i, \overline{x}_i]$ has $\underline{x}_i = 0$.

12. Results: Summary

Case	E	V	L, U	S
Narrow int.	$O(n)$	$O(n)$	$O(n \cdot \log(n))$	$O(n^2)$
Slightly wider narrow int.	$O(n)$	$O(n \cdot \log(n))$	$O(n \cdot \log(n))$	?
Single MI	$O(n)$	$O(n)$	$O(n \cdot \log(n))$	$O(n^2)$
Several MI	$O(n)$	$O(n^m)$	$O(n^m)$	$O(n^{2m})$
New case	$O(n)$	$O(n^m)$	?	?
Privacy case	$O(n)$	$O(n)$	$O(n \cdot \log(n))$	$O(n^2)$
Non-detects	$O(n)$	$O(n)$	$O(n \cdot \log(n))$	$O(n^2)$
General	$O(n)$	NP-hard	NP-hard	?

Here:

- S is skewness; $L = E - k_0 \cdot \sigma$ and $U = E + k_0 \cdot \sigma$ are endpoints of the confidence interval;
- the "new case" (described later) is a generalization of the case of several MI.

13. First Statistical Characteristic: Lower Bound $\underline{V}$ for the Range $[\underline{V}, \overline{V}]$ of Sample Variance V

- *First result:* the lower bound $\underline{V}$ can be always computed in time $O(n \cdot \log(n))$.
- *Second result:* a faster $O(n)$ algorithm.
- *Main idea:*
  - previously, an $O(n \cdot \log(n))$ sorting algorithm was used;
  - instead, we repeatedly use a linear-time $O(n)$ algorithm for computing the median.
- *Comment:*
  - we have developed a similar linear-time algorithm that computes $\overline{V}$ for several classes of problems;
  - later in this talk, we will present details of that algorithm.

14. Computing the Upper Endpoint $\overline{V}$ for the Range of Variance: Single MI Case and Its Subcases

- *What was known before:*
  - *In general:* computing $\overline{V}$ is NP-hard.
  - *Known $O(n^2)$ algorithm:* when intervals $[\underline{x}_i, \overline{x}_i] = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$ do not intersect.
  - *More general case:* "narrowed" intervals $[x_i^-, x_i^+] \stackrel{\text{def}}{=} [\widetilde{x}_i - \Delta_i/n, \widetilde{x}_i + \Delta_i/n]$ do not intersect.
  - *Known $O(n^2)$ algorithm:* for this case as well.
- *New result:*
  - *New case:* "narrowed" intervals $[x_i^-, x_i^+]$ satisfy a subset property: $[x_i^-, x_i^+] \not\subseteq (x_j^-, x_j^+)$.
  - *Particular cases:* narrow intervals, slightly wider narrow intervals, single MI, privacy case, no-detects.
  - *New algorithm:* computes $\overline{V}$ in linear time.

15. A Sample New Result: A Linear Algorithm for Computing Variance Under Interval Uncertainty

- *Given:* n intervals

$$\mathbf{x}_1 = [\widetilde{x}_n - \Delta_n, \widetilde{x}_1 + \Delta_1], \ldots, \mathbf{x}_n = [\widetilde{x}_n - \Delta_n, \widetilde{x}_n + \Delta_n].$$

- *Compute:* the upper endpoint $\overline{V}$ of the range

$$[\underline{V}, \overline{V}] = \left\{ \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^2 : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n \right\},$$

where $E \stackrel{\text{def}}{=} \frac{1}{n} \cdot \sum_{i=1}^{n} x_i.$

- *Known fact:* in general, this problem is NP-hard.
- *Our case:* $[x_i^-, x_i^+] \not\subseteq (x_j^-, x_j^+)$ for “narrowed” intervals

$$[x_i^-, x_i^+] \stackrel{\text{def}}{=} [\widetilde{x}_i - \Delta_i/n, \widetilde{x}_i + \Delta_i/n].$$

16. Towards a Linear-Time Algorithm: First Step

- *Known fact:* the function V is convex.
- *Geometric conclusion:* its maximum on a polytope $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$ is attained at its vertices.
- *Conclusion reformulated in algebraic terms:* for each i, we have $x_i = \underline{x}_i$ or $x_i = \overline{x}_i$.
- *Auxiliary result:*
  - if we sort intervals by their midpoints $\widetilde{x}_i$,
  - then, in the above case, the maximum is attained on one of the vectors $(\underline{x}_1, \ldots, \underline{x}_k, \overline{x}_{k+1}, \ldots, \overline{x}_n)$.
- *Intuitive explanation:* to maximize V, we "drag" all the points as far away from E as possible:
  - values $x_i < E$ are dragged to the left, to $\underline{x}_i$;
  - values $x_i > E$ are dragged to the right, to $\overline{x}_i$.

17. First Algorithm: $O(n^2)$

- *Natural algorithm:*
  - We sort intervals by their midpoints $\widetilde{x}_i$.
  - For each k from 0 to n, we compute

$$V(k) \stackrel{\text{def}}{=} V(\underline{x}_1, \ldots, \underline{x}_k, \overline{x}_{k+1}, \ldots, \overline{x}_n).$$

  - We choose the largest of computed $V(k)$'s as $\overline{V}$.
- *Time complexity:*
  - Sorting requires $O(n \cdot \log(n))$ steps.
  - Computing each $V(k)$ requires $O(n)$ steps, and computing $n+1$ different $V(k)$'s requires $O(n^2)$ steps.
  - Choosing the maximum requires $O(n)$ steps.
  - Totally, $O(n^2)$ steps.

18. Towards an $O(n \cdot \log(n))$ Algorithm

- *Most time-consuming stage:* computing all $n+1$ values of $V(k)$'s requires $(n+1) \cdot O(n) = O(n^2)$ steps.
- *Main idea:* use $V(k-1)$ to speed up computing $V(k)$.
- *Expression for $V(k)$:* $V(k) = M(k) - E(k)^2$, where

$$E(k) \stackrel{\text{def}}{=} \frac{1}{n} \cdot \left(\sum_{i=1}^{k} \underline{x}_i + \sum_{i=k+1}^{n} \overline{x}_i \right);$$

$$M(k) \stackrel{\text{def}}{=} \frac{1}{n} \cdot \left(\sum_{i=1}^{k} \underline{x}_i^2 + \sum_{i=k+1}^{n} \overline{x}_i^2 \right).$$

- *Corollary:*

$$E(k) = E(k-1) - \frac{1}{n} \cdot (\overline{x}_k - \underline{x}_k),$$

$$M(k) = M(k-1) - \frac{1}{n} \cdot (\overline{x}_k^2 - \underline{x}_k^2).$$

19. Resulting $O(n \cdot \log(n))$ Algorithm

- *First stage:* compute $E(0) = \frac{1}{n} \cdot \sum_{i=1}^{n} \overline{x}_i$,

$$M(0) = \frac{1}{n} \cdot \sum_{i=1}^{n} (\overline{x}_i)^2, \quad V(0) = M(0) - E(0)^2.$$

- For $k = 1$ to n, compute $E(k) = E(k-1) - \frac{1}{n} \cdot (\overline{x}_i - \underline{x}_i)$,

$$M(k) = M(k-1) - \frac{1}{n} \cdot (\overline{x}_i^2 - \underline{x}_i^2), \quad V(k) = M(k) - E(k)^2.$$

- Sorting requires $O(n \cdot \log(n))$ steps.
- Computing $V(0)$ requires $O(n)$ steps.
- Computing n values $V(1), \ldots, V(n)$ requires $n \cdot O(1) = O(n)$ steps.
- *Overall:* $\underline{O(n \cdot \log(n))} + O(n) + O(n) = O(n \cdot \log(n))$.

20. How to Avoid Sorting?

- *Most time-consuming stage:* sorting requires $O(n \cdot \log(n))$ steps.
- *Why we need sorting:* the formula
$$V(k) = V(\underline{x}_1, \ldots, \underline{x}_k, \overline{x}_{k+1}, \ldots, \overline{x}_n)$$
requires that intervals are already sorted by midpoints $\widetilde{x}_i$.
- *Objective:* compute $V(k)$ without sorting.
- *Idea:*
  - find the value of $\widetilde{x}_{(k)}$ (the k-th smallest midpoint);
  - divide indices of n intervals into two sets:
$$I^- = \{i : \ \widetilde{x}_i \le \widetilde{x}_{(k)}\}, \quad I^+ = \{i : \ \widetilde{x}_i > \widetilde{x}_{(k)}\}.$$
  - choose $x_i = \underline{x}_i$ if $i \in I^-$, and $x_i = \overline{x}_i$ if $i \in I^+$;
  - compute $V(k) = V(x_1, \ldots, x_n)$.
- We can compute $V(k)$ in $O(n)$ steps w/o sorting.

21. Decreasing the Number of Computed Values $V(k)$

+ No need for sorting, only $O(n)$ steps left.

– Since intervals are not sorted, we cannot compute $V(k)$ in terms of $V(k-1)$, so we need $n \times O(n)$ steps.

? Is it possible to compute $V(k)$ only for *some* k?

- *Lemma:*
  - first $V(k)$ increases: $V(k-1) < V(k)$;
  - $V(k)$ may stay maximum for several k's:

$$V(k-1) = V(k);$$

  - then $V(k)$ decreases: $V(k-1) > V(k)$.

- *Conclusion:* by comparing $V(k-1)$ with $V(k)$, we can tell whether we are to the left or to the right of $k_{\max}$.

- *Approach:* we can use binary search to find the optimal value of k.

22. Further Simplification

- *Simplifying Lemma:*
  - $V(k-1) < V(k)$ if and only if $\widetilde{x}_{(k)} - \Delta_{(k)}/n < E(k)$;
  - $V(k-1) > V(k)$ if and only if $\widetilde{x}_{(k)} - \Delta_{(k)}/n > E(k)$;
  - $V(k-1) = V(k)$ if and only if $\widetilde{x}_{(k)} - \Delta_{(k)}/n = E(k)$.
- *Remaining problem:*
  - since we use binary search, we need to compare $(V(k-1)$ and $V(k))$ $O(\log(n))$ times;
  - we need to compute $O(\log(n))$ different values

$$\widetilde{x}_{(k)} - \Delta_{(k)}/n \text{ and } E(k);$$

  - finding $\widetilde{x}_{(k)}$ (the k-th smallest midpoint) requires $O(n)$ steps;
  - so, overall, we still need

$$O(\log(n)) \cdot O(n) = O(n \cdot \log(n)) \text{ steps.}$$

23. Towards a Final Speed-up

At each iteration of binary search

- *We know:*
  - l and g such that $l \le k_{\max} \le g$;
  - $\widetilde{x}_{(l)}$, sets $I_l^- = \{i : \ \widetilde{x}_i \le \widetilde{x}_{(l)}\}$, $I_l^+ = \{i : \ \widetilde{x}_i > \widetilde{x}_{(l)}\}$;
  - $\widetilde{x}_{(g)}$, sets $I_g^- = \{i : \widetilde{x}_i \le \widetilde{x}_{(g)}\}$, $I_g^+ = \{i : \widetilde{x}_i > \widetilde{x}_{(g)}\}$;
  - $\widetilde{x}_{(l)} - \Delta_{(l)}/n$, $\widetilde{x}_{(g)} - \Delta_{(g)}/n$, $E(l)$ and $E(g)$.
- *We compute:*
  - values $m = \lfloor \dfrac{l+g}{2} \rfloor$, $\widetilde{x}_{(m)}$,
  - sets $I_m^- = \{i : \widetilde{x}_i \le \widetilde{x}_{(m)}\}$ and $I_m^+ = \{i : \widetilde{x}_i > \widetilde{x}_{(m)}\}$,
  - values $\widetilde{x}_{(m)} - \Delta_{(m)}/n$ and $E(m)$.
- *Idea:* use what is known for l and g to speed up the computations related to m.

24. Final Idea

$[\quad I_l^- \quad][\qquad\qquad I_l^+ \qquad\qquad]$

$[\qquad\qquad I_g^- \qquad\qquad][\quad I_g^+ \quad]$

$[\qquad I_m^- \qquad][\qquad I_m^+ \qquad]$

- By definition, $I_l^+ \cap I_g^- = \{i : \widetilde{x}_{(l)} \le \widetilde{x}_i < \widetilde{x}_{(g)}\}$.
- *Observation:* $\widetilde{x}_{(m)}$ is the median of the midpoints indexed by indices in $I_l^+ \cap I_g^-$.
- We can compute m and $\widetilde{x}_{(m)} - \Delta_{(m)}/n$ in time $O(g-l)$.
- *Fact:* $I_l^- \subseteq I_m^-$ and $I_g^+ \subseteq I_m^+$.
- *Idea:* we can use $x_{(m)}$ to divide $I_l^+ \cap I_g^-$ into two sets P^- and P^+ such that

$$I_m^- = I_l^- \cup P^- \text{ and } I_m^+ = I_g^+ \cup P^+.$$

25. Final Algorithm

At each iteration, we have:

- $I^- = \{i : \text{we know that } x_{\max,i} = \underline{x}_i\}$; initially, $I^- = \emptyset$;
- $I^+ = \{i : \text{we know that } x_{\max,i} = \overline{x}_i\}$; initially, $I^+ = \emptyset$;
- $I \stackrel{\text{def}}{=} \{1, \ldots, n\} - I^- - I^+$, $E^- \stackrel{\text{def}}{=} \sum\limits_{i \in I^-} \underline{x}_i$, $E^+ \stackrel{\text{def}}{=} \sum\limits_{j \in I^+} \overline{x}_j$.

At each iteration, we do the following:

- compute the median m of I (in terms of sorting by $\widetilde{x}_i$);
- divide I into $P^- = \{i : \widetilde{x}_i \le \widetilde{x}_m\}$, $P^+ = \{j : \widetilde{x}_j > \widetilde{x}_m\}$;
- compute $e^- = E^- + \sum\limits_{i \in P^-} \underline{x}_i$ and $e^+ = E^+ + \sum\limits_{j \in P^+} \overline{x}_j$;
- if $n \cdot x^-_m < e^- + e^+$: $I^- := I^- \cup P^-$, $E^- := e^-$, $I := P^+$;
- if $n \cdot x^-_m > e^- + e^+$: $I^+ := I^+ \cup P^+$, $E^+ := e^+$, $I := P^-$;
- otherwise: $I^- := I^- \cup P^-$, $I^+ := I^+ \cup P^+$, $I := \emptyset$.

26. New Algorithm Requires Linear Time: Proof

- *At each iteration:*
  - *computing median* requires linear time: $t \le C_1 \cdot |I|$ for some C_1;
  - *all other operations* with I also require linear time: $t \le C_2 \cdot |I|$ for some C_2;
  - *conclusion:* iteration time is: $t \le C \cdot |I|$, where $C \stackrel{\text{def}}{=} C_1 + C_2$.
- *We start:* with the set I of size n.
- *Then:* we have a set I of size $\dfrac{n}{2}$, of size $\dfrac{n}{4}$, etc.
- *Result:* the overall computation time is

$$\le C \cdot \left(n + \frac{n}{2} + \frac{n}{4} + \ldots\right) \le C \cdot 2n.$$

- *Conclusion:* the new algorithm requires linear time.

27. Computing Upper Bound for the Variance: New Case

- *Corollary:* an $O(n^m)$ algorithm exists for m MI.
- *New case:* for some $m \geq 1$ and $K \geq 2$, the intervals $[\underline{x}_i, \overline{x}_i]$ can be divided into m subclasses $I_1, \ldots, I_m$ s. t.:
  - within each I_j $(j < m)$ no narrowed interval
  $$[x_i^-, x_i^+] = [\widetilde{x}_i - \Delta_i/n, \widetilde{x}_i + \Delta_i/n]$$
  is a proper subset of another one: $[x_i^-, x_i^+] \not\subseteq (x_{i'}^-, x_{i'}^+)$;
  - I_m either has the same property, or
  $$[x_{i_1}^-, x_{i_1}^+] \cap \ldots \cap [x_{i_K}^-, x_{i_K}^+] = \emptyset$$
  for every K different narrowed intervals from I_m.
- *Observation:* this is a generalization of the case of m MI.
- *New result:* we have designed an $O(n^m)$ algorithm for the new case.

28. Computing Range for Outliers Detection: Results

- *Detecting outliers:* x_i is an outlier if $x_i \notin [L, U]$, where $L \stackrel{\text{def}}{=} E - k_0 \cdot \sqrt{V}$, $U \stackrel{\text{def}}{=} E + k_0 \cdot \sqrt{V}$.
- *First results:*
  - $O(n \cdot \log(n))$ algorithms for computing $\overline{L}$ and $\underline{U}$;
  - computing $\underline{L}$ and $\overline{U}$ is NP-hard;
  - $O(n^2)$ algorithms for computing $\underline{L}$ and $\overline{U}$ when K "narrowed" intervals $\left[\widetilde{x}_i - \Delta_i \cdot \frac{1+\alpha^2}{n}, \widetilde{x}_i + \Delta_i \cdot \frac{1+\alpha^2}{n}\right]$ have an empty intersection.
- *Faster algorithms:*
  - $O(n \cdot \log(n))$ algorithms for computing $\underline{L}$ and $\overline{U}$ in the above case and in the single MI case;
  - $O(n^m)$ algorithms for computing $\underline{L}$ and $\overline{U}$ for the case of m MI.

29. Computing the Range for Skewness under Interval Uncertainty

- *Skewness – reminder:*

$$S(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^3.$$

- *Practical importance:* S is a measure of the distribution's asymmetry.
- *Given:* n intervals $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1], \ldots, [\underline{x}_n, \overline{x}_n]$.
- *Compute:* the range

$$[\underline{S}, \overline{S}] \stackrel{\text{def}}{=} \{S(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

- *First result:* $O(n^2)$ algorithms for computing $\underline{S}$ and $\overline{S}$ in the case of single MI (and its subcases).
- *Second result:* $O(n^{2m})$ algorithms for computing $\underline{S}$ and $\overline{S}$ in the case of m MIs.

30. Application to Radar Data Processing

- *Situation:* a radar observes the result of an explosion.
- *Practical problem:* distinguish between the core and the slowly out-moving fragments of the explosion.
- *Specifics:*
  - due to radar's low horizontal resolution, we get a 1-D signal $x(t)$ representing different 2-D slices;
  - this corresponds to intervals of distance.
- *Resulting problem:* combines two types of uncertainty:
  - interval uncertainty in distance, and
  - probabilistic uncertainty of measurement.
- *Our work:* adjust our techniques to this problem.

31. Formulation of the Problem

- *Problem:* Identify the core of the result of space explosion (e.g., supernovae, planet destruction).
- Space explosions are important, because, e.g., supernovae explosions is how heavy metals spread around in the Universe.
- Explosions are rarely directly observed because they are rare and fast.
- *What we observe:*
  - the explosion core
    (the remainder of the original celestial body)
  - surrounded by the fragments.
- *Example:* Crab Nebula was formed after the 1054 supernovae explosion.

32. Formulation of the Problem (cont-d)

- In general, we have a 2-D (sometimes 3-D) image of the result of the explosion. In such cases, image processing techniques can detect the core.
- There is one important case when only 1-D information is available: radar observations, the main source of information
- A radar sends a pulse signal toward an object, this signal reflects from the object back to the station; and we measure the travel time t.
- So, we know the distance $d = c \cdot t/2$ to the object.
- It is difficult to separate the signals from different fragments located at the same distance.
- Hence, we observe a 1-D signal $s(t)$ = the total intensity of all the fragments at distance $c \cdot t/2$.

33. A New Method for Solving the Problem: Main Idea

- *At first glance:* there is no difference between the signals from the fragments and the core.
- *Idea:* after the explosion, fragments usually start rotating fast.
- *Comment:* they rotate at random rotation frequencies, with random phases.
- *Conclusion:*
  - signals from the fragments oscillate, while
  - the signal from the core practically does not change.
- *Resulting idea:*
  - measure $s(t)$ at several consequent moments of time $T_1 < \ldots < T_N$, and
  - use the above difference to identify the core.

34. The Corresponding t-Scales are Linearly Related

- *Problem:* we must compare signals measured at different times $T_k \neq T_l$.
- Let's use coordinates where radar is at (0,0), x-axis directed towards "cloud".
- Let T_0 be the moment of explosion, let $x_0 \stackrel{\text{def}}{=} x(T_0)$.
- Since there is no friction in space, $x^{(i)}(T_k) = x_0 + v_x^{(i)} \cdot (T_k - T_0)$. So, radar signals at moments T_k and T_l are: $t_k^{(i)} = \dfrac{x_0}{c} + v_x^{(i)} \cdot \dfrac{T_k - T_0}{c}$ (same for $t_l^{(i)}$).
- Hence, $t_l^{(i)} = a_{kl} \cdot t_k^{(i)} + b_{kl}$, where $a_{kl} = \dfrac{T_l - T_0}{T_k - T_0} > 0$ and $b_{kl} = \dfrac{x_0}{c} - \dfrac{x_0}{T_k - T_0} \cdot \dfrac{T_l - T_0}{c}$ are the same for all i.
- *Conclusion:* t-scales of the signals $s_k(t)$ and $s_l(t)$ are linearly related: $t_k \to t_l = a_{kl} \cdot t_k + b_{kl}$.

35. How Can We Experimentally Find the Coefficients of This Linear Relation?

- *Main idea:* by tracing the borders of the cloud.
- Let $\underline{t}_k$ be the smallest time at which we get some reflection from the fragments cloud.
- Let $\overline{t}_k$ be the largest time at which we observe the radar reflection from this cloud.
- *Reminder:* t_k and t_l are linearly related, with $a_{kl} > 0$.
- *Conclusion:* t_l is the smallest (largest) for the same fragment i for which t_k was the smallest (corr., largest):

$$\underline{t}_l = a_{kl} \cdot \underline{t}_k + b_{kl}; \quad \overline{t}_l = a_{kl} \cdot \overline{t}_k + b_{kl}.$$

- *Resulting algorithm:*

$$a_{kl} = \frac{\overline{t}_l - \underline{t}_l}{\overline{t}_k - \underline{t}_k}; \quad b_{kl} = \frac{\overline{t}_k \cdot \underline{t}_l - \underline{t}_k \cdot \overline{t}_l}{\overline{t}_k - \underline{t}_k}.$$

36. How Can We Transform Signals $s_k(t)$ and $s_l(t)$ to the Same Scale?

- *We know:* $s_k(t)$ describes the same fragment(s) as $s_l(t')$, where $t' = a_{kl} \cdot t + b_{kl}$.
- *Problem:* due to finite temporal resolution Δt (interval uncertainty), each $s_l(i \cdot \Delta)$ represents the entire "bin"

$$I_i \stackrel{\text{def}}{=} [(i - 0.5) \cdot \Delta t, (i + 0.5) \cdot \Delta t].$$

- *Physical meaning:* from T_k to T_l, the cloud expands.
- *Corollary:* fragments that were in the same bin I_j at T_k may be in different bins $\widetilde{I}_i \neq \widetilde{I}_{i'}$ at time T_l.
- *How can we match:* use linear interpolation

$$\widetilde{s}_l(i \cdot \Delta t) \stackrel{\text{def}}{=} \sum_j \frac{\|\widetilde{I}_i \cap I_j\|}{\Delta t} \cdot s_l(j \cdot \Delta)$$

- We will assume that the signals were thus rescaled.

37. Algorithm: Main Idea

- *Case 1:* bin contains $n(t)$ independent oscillated fragments (but no core).
- We assume that fragments are independent, hence the mean $E(t)$ in the bin t is $E(t) \approx n(t) \cdot E$, where E is the average over all bins.
- Similarly, for variance, $V(t) \approx n(t) \cdot V$, so
$$E(t) - (E/V) \cdot V(t) \approx 0.$$
- *Case 2:* bin also contains core, with intensity E_c.
- The core isn't rotating, so its variance is negligible.
- Hence, $E(t) \approx E_c + N(t) \cdot E$, $V(t) \approx N(t) \cdot V$, so
$$E(t) - (E/V) \cdot V(t) \approx E_c.$$
- *Intuitive idea:* find E/V, and the core is where
$$E(t) - (E/V) \cdot V(t) \to \max_t .$$

38. Towards a Statistical Algorithm

- The intensity $I_i(t)$ of i-th fragment depends on time.
- $a_i \stackrel{\text{def}}{=} \lim\limits_{T\to\infty} T^{-1}\cdot\int\limits_0^T I_i(t)\,dt$, $b_i \stackrel{\text{def}}{=} \lim\limits_{T\to\infty} T^{-1}\cdot\int\limits_0^T (I_i(t)-a_i)^2\,dt$.
- $a_0 \stackrel{\text{def}}{=} E[a_i]$, $b_0 \stackrel{\text{def}}{=} E[b_i]$, $A_0 \stackrel{\text{def}}{=} V[a_i]$, $B_0 \stackrel{\text{def}}{=} V[b_i]$.
- Due to Central Limit Theorem, distribution is normal:

$$\rho = \prod_{t=1}^N \frac{1}{\sqrt{2\pi\cdot n(t)\cdot A_0}}\cdot \exp\left(-\frac{(E(t)-n(t)\cdot a_0)^2}{2n(t)\cdot A_0}\right)\times$$

$$\prod_{t=1}^N \frac{1}{\sqrt{2\pi\cdot n(t)\cdot B_0}}\cdot \exp\left(-\frac{(V(t)-n(t)\cdot b_0)^2}{2n(t)\cdot B_0}\right).$$

- For the layer $t=t_0$ containing the core, we have $E(t)-E_c-n(t)\cdot a_0$ instead of $E(t)-n(t)\cdot a_0$.
- *Objective:* based on $E(t)$ and $V(t)$, find t_0 by using the Maximum Likelihood Method $\psi \stackrel{\text{def}}{=} -\ln(\rho) \to \min$.

39. Resulting Algorithm

- *Algorithm:*
  - Re-scale the signals $s_k(t)$ into $\widetilde{s}_k(t)$ so that the same value t corresponds to the same fragments.
  - For each t, we compute the sample average $E(t)$ and the sample variance $V(t)$ of the values $\widetilde{s}_k(t)$.
  - For each t, we compute v_t and $\psi_0(t)$.
  - Find t_0 for which $\psi_0(t_0) = m \stackrel{\text{def}}{=} \max\limits_t \psi_0(t)$.
- *How reliable is this estimate?*
  - with reliability 95%, the core is among those t for which $\psi_0(t) \geq m - 2$ (this is 2σ interval);
  - with reliability 99.9%, the core is among those t for which $\psi_0(t) \geq m - 4.5$ (this is 3σ interval).

40. Application to Geosciences

- *Objective:* find the structure of the Earth.
- *Typical algorithm*– Hole's code:
  - *observe* the traveltimes t_i, and
  - *find* velocities v_j for which $t_i = \sum\limits_j \dfrac{\ell_{ij}}{v_j}$.
- *Problem:* the resulting velocities $\widetilde{v}_j$ are sometimes unphysical.
- *Idea:* we often know bounds $[\underline{v}_j, \overline{v}_j]$ on v_j.
- *Mathematical problem:* solve the above seismic inverse problem under this interval uncertainty.
- *Additional problem:* in addition to interval uncertainty, we must take into account probabilistic uncertainty.
- *Our result:* adjusted general techniques for combining interval and probabilistic uncertainty to this problem.

41. Application to Computer Engineering: Chip Design

- *Main objective:* decrease the clock cycle D.
- *Current approach:* worst-case (interval) techniques, i.e.,

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

  where $D_i = \sum\limits_{j=1}^{n} a_{ij} \cdot x_j$ is the delay along the i-th path.
- *Problem:* the probability of the combination of worst-case values is extremely small.
- *Result:* over-conservative estimates, leading to unnecessary over-design and under-performance of circuits.
- *Additional information:* we often have *partial* information about probability distributions of x_j.
- *Our result:* produced estimates which are valid for all distributions consistent with this information.

42. Conclusions and Future Work

- *Statistical analysis* is practically important.
- *Traditionally:* it is assumed that we know the exact values $x_1, \ldots, x_n$.
- *In practice:* interval uncertainty $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$.
- *Resulting problem:* given intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$, compute the range $\mathbf{C}$ of $C(x_1, \ldots, x_n)$ when $x_i \in \mathbf{x}_i$.
- *Known:* NP-hard in general, $O(n^2)$ algorithms known for some cases.
- *Our main results:* we reduced the computational complexity to $O(n \cdot \log(n))$ and $O(n)$.
- *Applications:* computer security, geoinformatics, chip design, radar data processing, etc.
- *Remaining problems:* faster algorithms, new C, taking partial information about probabilities into account.

43. Acknowledgments

This work was supported in part by:

- by the Japan Advanced Institute of Science and Technology (JAIST) International Joint Research Grant 2006-08,
- by Texas Department of Transportation contract No. 0-5453,
- by National Science Foundation grants HRD-0734825, EAR-0225670, and EIA-0080940, and
- and by the Max Planck Institut für Mathematik.