# Estimating Variance under Interval and Fuzzy Uncertainty: Parallel Algorithms

Karen Villaverde and Gang Xiang

***Abstract*—Traditional data processing in science and engineering starts with computing the basic statistical characteristics such as the population mean $E$ and population variance $V$. In computing these characteristics, it is usually assumed that the corresponding data values $x_1, \ldots, x_n$ are known exactly. In many practical situations, we only know intervals $[\underline{x}_i, \overline{x}_i]$ that contain the actual (unknown) values of $x_i$ or, more generally, a fuzzy number that describes $x_i$. In this case, different possible values of $x_i$ lead, in general, to different values of $E$ and $V$. In such situations, we are interested in producing the intervals of possible values of $E$ and $V$ – or fuzzy numbers describing $E$ and $V$. There exist algorithms for producing such interval and fuzzy estimates. However, these algorithms are more complex than the typical data processing formulas and thus, require a larger amount of computation time. If we have several processors, then, it is desirable to perform these algorithms in parallel on several processors, and thus, to speed up computations. In this paper, we show how the algorithms for estimating variance under interval and fuzzy uncertainty can be parallelized.**

## I. Computing statistics is important

Traditional data processing in science and engineering starts with computing the basic statistical characteristics such as the population mean

$$E = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

and population variance

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

## II. Additional problem

Traditional engineering statistical formulas assume that we know the *exact* values $x_i$ of the corresponding quantities. In practice, these values come either from measurements or from expert estimates. In both cases, we get only *approximations* $\widetilde{x}_i$ to the actual (unknown) values $x_i$.

When we use these approximate values $\widetilde{x}_i \neq x_i$ to compute the desired statistical characteristics such as $E$ and $V$, we only get approximate valued $\widetilde{E}$ and $\widetilde{V}$ for these characteristics. It is desirable to estimate the accuracy of these approximations.

Karen Villaverde is with the Department of Computer Science, New Mexico State University, email kvillave@cs.nmsu.edu, and Gang Xiang is with Philips Healthcare Informatics, El Paso, Texas, email gxiang@acm.org.

This work was supported in part by an internal grant from New Mexico State University, by NSF grants HRD-0734825, EAR-0225670, and EIA-0080940, and by Texas Department of Transportation grant No. 0-5453. The authors are thankful to the anonymous referees for valuable suggestions.

## III. Case of measurement uncertainty

Measurements are never 100% accurate. As a result, the result $\widetilde{x}$ of the measurement is, in general, different from the (unknown) actual value $x$ of the desired quantity. The difference $\Delta x \stackrel{\text{def}}{=} \widetilde{x} - x$ between the measured and the actual values is usually called a *measurement error*.

The manufacturers of a measuring device usually provide us with an upper bound $\Delta$ for the (absolute value of) possible errors, i.e., with a bound $\Delta$ for which we guarantee that $|\Delta x| \leq \Delta$. The need for such a bound comes from the very nature of a measurement process: if no such bound is provided, this means that the difference between the (unknown) actual value $x$ and the observed value $\widetilde{x}$ can be as large as possible.

Since the (absolute value of the) measurement error $\Delta x = \tilde{x} - x$ is bounded by the given bound $\Delta$, we can therefore guarantee that the actual (unknown) value of the desired quantity belongs to the interval $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$.

## IV. Traditional probabilistic approach to describing measurement uncertainty

In many practical situations, we not only know the interval $[-\Delta, \Delta]$ of possible values of the measurement error; we also know the probability of different values $\Delta x$ within this interval [13].

In practice, we can determine the desired probabilities of different values of $\Delta x$ by comparing the results of measuring with this instrument with the results of measuring the same quantity by a standard (much more accurate) measuring instrument. Since the standard measuring instrument is much more accurate than the one used, the difference between these two measurement results is practically equal to the measurement error; thus, the empirical distribution of this difference is close to the desired probability distribution for measurement error.

## V. Interval approach to measurement uncertainty

As we have mentioned, in many practical situations, we do know the probabilities of different values of the measurement error. There are two cases, however, when this determination is not done:

- First is the case of cutting-edge measurements, e.g., measurements in fundamental science. When a Hubble telescope detects the light from a distant galaxy, there is no "standard" (much more accurate) telescope floating nearby that we can use to calibrate the Hubble: the Hubble telescope is the best we have.

- The second case is the case of measurements on the shop floor. In this case, in principle, every sensor can be thoroughly calibrated, but sensor calibration is so costly – usually costing ten times more than the sensor itself – that manufacturers rarely do it.

In both cases, we have no information about the probabilities of $\Delta x$; the only information we have is the upper bound on the measurement error.

In this case, after performing a measurement and getting a measurement result $\widetilde{x}$, the only information that we have about the actual value $x$ of the measured quantity is that it belongs to the interval $\mathbf{x} = [\widetilde{x} - \Delta, \widetilde{x} + \Delta]$. In this situation, for each $i$, we know the interval $\mathbf{x}_i$ of possible values of $x_i$, and we need to find the ranges $\mathbf{E}$ and $\mathbf{V}$ of the characteristics $E$ and $V$ over all possible tuples $x_i \in \mathbf{x}_i$.

## VI. Case of expert uncertainty

An expert usually describes his/her uncertainty by using words from the natural language, like "most probably, the value of the quantity is between 6 and 7, but it is somewhat possible to have values between 5 and 8". To formalize this knowledge, it is natural to use *fuzzy set theory*, a formalism specifically designed for describing this type of informal ("fuzzy") knowledge [9], [12].

As a result, for every value $x_i$, we have a fuzzy set $\mu_i(x_i)$ which describes the expert's prior knowledge about $x_i$: the number $\mu_i(x_i)$ describes the expert's degree of certainty that $x_i$ is a possible value of the $i$-th quantity.

An alternative user-friendly way to represent a fuzzy set is by using its $\alpha$-cuts $\{x_i \,|\, \mu_i(x_i) > \alpha\}$ (or $\{x_i \,|\, \mu_i(x_i) \geq \alpha\}$). For example, the $\alpha$-cut corresponding to $\alpha = 0$ is the set of all the values which are possible at all, the $\alpha$-cut corresponding to $\alpha = 0.1$ is the set of all the values which are possible with degree of certainty at least 0.1, etc. In these terms, a fuzzy set can be viewed as a nested family of intervals $[\underline{x}_i(\alpha), \overline{x}_i(\alpha)]$ corresponding to different level $\alpha$.

## VII. Estimating statistics under fuzzy uncertainty: precise formulation of the problem

In general, we have fuzzy knowledge $\mu_i(x_i)$ about each value $x_i$; we want to find the fuzzy set corresponding to a given characteristic $y = C(x_1, \ldots, x_n)$. Intuitively, the value $y$ is a reasonable value of the characteristic if $y = f(x_1, \ldots, x_n)$ for some reasonable values $x_i$, i.e., if for some values $x_1, \ldots, x_n$, $x_1$ is reasonable, and $x_2$ is reasonable, ..., and $y = f(x_1 \ldots, x_n)$. If we interpret "and" as $\min$ and "for some" ("or") as $\max$, then we conclude that the corresponding degree of certainty $\mu(y)$ in $y$ is equal to

$$\mu(y) =$$

$$\max\{\min(\mu_1(x_1), \ldots, \mu_n(x_n)) | C(x_1, \ldots, x_n) = y\}.$$

## VIII. Reduction to the case of interval uncertainty

It is known that the above formula (called *extension principle*) can be reformulated as follows: for each $\alpha$, the $\alpha$-cut $\mathbf{y}(\alpha)$ of $y$ is equal to the range of possible values of $C(x_1, \ldots, x_n)$ when $x_i \in \mathbf{x}_i(\alpha)$ for all $i$. Thus, from the computational viewpoint, the problem of computing the statistical characteristic under fuzzy uncertainty can be reduced to the problem of computing this characteristic under interval uncertainty; see, e.g., [5]

In view of this reduction, in the following text, we will consider the case of interval uncertainty.

## IX. Estimating statistics under interval uncertainty: a problem

In the case of interval uncertainty, instead of the true values $x_1, \ldots, x_n$, we only know the intervals $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1], \ldots, \mathbf{x}_n = [\underline{x}_n, \overline{x}_n]$ that contain the (unknown) true values of the measured quantities. For different values $x_i \in \mathbf{x}_i$, we get, in general, different values of the corresponding statistical characteristic $C(x_1, \ldots, x_n)$. Since all values $x_i \in \mathbf{x}_i$ are possible, we conclude that all the values $C(x_1, \ldots, x_n)$ corresponding to $x_i \in \mathbf{x}_i$ are possible estimates for the corresponding statistical characteristic. Therefore, for the interval data $\mathbf{x}_1, \ldots, \mathbf{x}_n$, a reasonable estimate for the corresponding statistical characteristic is the range

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

We must therefore modify the existing statistical algorithms so that they compute, or bound these ranges.

## X. Estimating mean under interval uncertainty

The arithmetic average $E$ is a monotonically increasing function of each of its $n$ variables $x_1, \ldots, x_n$, so its smallest possible value $\underline{E}$ is attained when each value $x_i$ is the smallest possible ($x_i = \underline{x}_i$) and its largest possible value is attained when $x_i = \overline{x}_i$ for all $i$. In other words, the range $\mathbf{E}$ of $E$ is equal to $[E(\underline{x}_1, \ldots, \underline{x}_n), E(\overline{x}_1, \ldots, \overline{x}_n)]$. In other words, $\underline{E} = \dfrac{1}{n} \cdot (\underline{x}_1 + \ldots + \underline{x}_n)$ and $\overline{E} = \dfrac{1}{n} \cdot (\overline{x}_1 + \ldots + \overline{x}_n)$.

## XI. Estimating variance under interval uncertainty

It is known that the problem of computing the exact range $\mathbf{V} = [\underline{V}, \overline{V}]$ for the variance $V$ over interval data $x_i \in [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$ is, in general, NP-hard; see, e.g., [10], [11]. Specifically, there is a $O(n \cdot \log(n))$ time algorithm for computing $\underline{V}$, but computing $\overline{V}$ is, in general, NP-hard.

In many practical situations, there are efficient algorithms for computing $\overline{V}$: e.g., an $O(n \cdot \log(n))$ time algorithm exists when no two narrowed intervals $[x_i^-, x_i^+]$ (where $x_i^- \stackrel{\text{def}}{=} \widetilde{x}_i - \dfrac{\Delta_i}{n}$ and $x_i^+ \stackrel{\text{def}}{=} \widetilde{x}_i + \dfrac{\Delta_i}{n}$) are proper subsets of one another, i.e., when $[x_i^-, x_i^+] \not\subseteq (x_j^-, x_j^+)$ for all $i$ and $j$ [4].

## XII. Comment about the possibility of linear-time algorithms

As we will see, in the $O(n \cdot \log(n))$ algorithm, the main computation time is used on sorting. It is possible to avoid sorting when estimating variance under interval uncertainty (see, e.g., [6], [15]), and use instead the known fact that we can compute the median of a set of $n$ elements in linear time (see, e.g., [3]). (This use of median is similar to the one from [2], [7].)

It is worth mentioning, however, that while asymptotically, the linear time algorithm for computing the median is faster than sorting, this median computing algorithm is still rather complex – so, for reasonable size $n$, sorting is faster than computing the median – and thus, sorting-based algorithms are actually faster than median-based ones.

## XIII. Need for parallelization

Traditional algorithms for computing the population variance $V$ based on the exact values $x_1, \ldots, x_n$ require linear time $O(n)$. Algorithms for estimating variance under interval uncertainty require a larger amount of computation time – e.g., time $O(n \cdot \log(n))$. How can we speed up these computations?

If we have several processors, then it is desirable to perform these algorithms in parallel on several processors, and thus, speed up computations. In this paper, we show how the algorithms for estimating variance under interval and fuzzy uncertainty can be parallelized.

In order to describe how to parallelize these algorithms, let us describe the existing sequential (non-parallel) algorithms for estimating the variance under interval uncertainty.

## XIV. Algorithm for computing $\overline{V}$ in the no-proper-subset case

The algorithm from [4] is as follows:

- First, we sort the values $\widetilde{x}_i$ into an increasing sequence. Without losing generality, we can assume that
$$\widetilde{x}_1 \le \widetilde{x}_2 \le \ldots \le \widetilde{x}_n.$$
- Then, for every $k$ from 0 to $n$, we compute the value $V^{(k)} = M^{(k)} - (E^{(k)})^2$ of the population variance $V$ for the vector $x^{(k)} = (\underline{x}_1, \ldots, \underline{x}_k, \overline{x}_{k+1}, \ldots, \overline{x}_n)$. (For $k = 0$, $x^{(0)} = (\overline{x}_1, \ldots, \overline{x}_n)$.)
- Finally, we compute $\overline{V}$ as the largest of $n+1$ values $V^{(0)}, \ldots, V^{(n)}$.

To compute the values $V^{(k)}$, first, we explicitly compute $M^{(0)} = \dfrac{1}{n} \cdot \sum\limits_{i=1}^{n} (\overline{x}_i)^2$, $E^{(0)} = \dfrac{1}{n} \cdot \sum\limits_{i=1}^{n} \overline{x}_i$, and
$$V^{(0)} = M^{(0)} - (E^{(0)})^2.$$
Once we know the values $M^{(k)}$ and $E^{(k)}$, we can compute
$$M^{(k+1)} = M^{(k)} + \frac{1}{n} \cdot (\underline{x}_{k+1})^2 - \frac{1}{n} \cdot (\overline{x}_{k+1})^2$$
and
$$E^{(k+1)} = E^{(k)} + \frac{1}{n} \cdot \underline{x}_{k+1} - \frac{1}{n} \cdot \overline{x}_{k+1}.$$

## XV. Possibility of parallelization

For large $n$, we may want to further speed up computations if we have several processors working in parallel.

In the general case, all the stages of the above algorithm can be parallelized by known techniques. In particular, Stage 3 is a particular case of a general *prefix-sum* problem, in which we must compute the values
$$a_n, \quad a_n * a_{n-1}, \quad a_n * a_{n-1} * a_{n-2}, \ldots,$$
for some associative operation $*$ (in our case, $* = \max$).

## XVI. Case of potentially unlimited number of processors

If we have a potentially unlimited number of processors, then we can do the following (see, e.g., [8], for the information on how to parallelize the corresponding stages):

- on Stage 1, we can sort the values $\widetilde{x}_i$ in time $O(\log(n))$;
- on Stage 2, we can compute the values $V^{(i)}$ (i.e., solve the prefix-sum problem) in time $O(\log(n))$;
- on Stage 3, we can compute the maximum of $V^{(i)}$ in time $O(\log(n))$.

As a result, we can check monotonicity in time
$$O(\log(n)) + O(\log(n)) + O(\log(n)) = O(\log(n)).$$

## XVII. Example

To give the readers a better understanding on how these stages can be parallelized, let us describe, in detail, parallelization of Stage 3. In other words, let us describe how to compute the maximum of $n+1$ given values $V^{(0)}, \ldots, V^{(n)}$ in parallel.

As we have mentioned, the parallelized algorithm consists of $O(\log(n))$ steps. At the first step, we divide $n+1$ values into pairs $(V^{(0)}, V^{(1)})$, $(V^{(2)}, V^{(3)})$, ... Since we have assumed that we have a potentially unlimited number of processors, we can allocate an individual processor to each pair – to the total of $\lceil (n+1)/2 \rceil$ processors. At the first step, each processor compares the corresponding two numbers and thus computes the maximum of this pair:

- the first processor computes the value
$$m(0,1) \stackrel{\text{def}}{=} \max(V^{(0)}, V^{(1)});$$
- at the same time, the second processor computes the value
$$m(2,3) \stackrel{\text{def}}{=} \max(V^{(2)}, V^{(3)});$$
- etc.

At the end of the first step, we thus have $\lceil (n+1)/2 \rceil \approx n/2$ values $m(0,1)$, $m(2,3)$, $m(4,5)$, $m(6,7)$, etc.

At the second step, we divide these $\lceil (n+1)/2 \rceil \approx n/2$ values into pairs, and compute the maximum of each pair:

- the first processor computes the value
$$m(0,3) \stackrel{\text{def}}{=} \max(m(0,1), m(2,3));$$
by definition of $m(0,1)$ and $m(2,3)$, this value is equal to $\max(V^{(0)}, V^{(1)}, V^{(2)}, V^{(3)})$;

- at the same time, the second processor computes the value
$$m(4,7) \stackrel{\text{def}}{=} \max(m(4,5), m(6,7));$$
by definition of $m(4,5)$ and $m(6,7)$, this value is equal to $\max(V^{(4)}, V^{(5)}, V^{(6)}, V^{(7)})$;
- etc.

At the end of the second step, we thus have $\approx n/4$ values $m(0,3)$, $m(4,7)$, etc., describing the maxima of four elements.

At the third step, we repeat this procedure again, and get the values $m(0,7)$, $m(8,15)$, etc., describing the maxima of $8 = 2^3$ elements.

At the $k$-th step, we get the values
$$m(0, 2^k - 1), m(2^k, 2^k + (2^k - 1)), \ldots,$$
describing the maxima of $2^k$ elements.

As soon as we get $2^k = n$, i.e., as soon as $k \approx \log_2(n)$, we get the desired maximum of all $n$ elements. Thus, we can indeed compute the desired maximum in $O(\log(n))$ steps.

## XVIII. Case of a fixed number of processors

If we have $p < n$ processors, then we can:

- on Stage 1, sort $n$ values in time
$$O\left(\frac{n \cdot \log(n)}{p} + \log(n)\right);$$
see, e.g., [8];
- on Stage 2, compute the values $V^{(i)}$ in time
$$O\left(\frac{n}{p} + \log(p)\right);$$
see, e.g., [1];
- on Stage 3, compute the maximum of $V^{(i)}$ in time
$$O\left(\frac{n}{p} + \log(p)\right).$$

Overall, we thus need time
$$O\left(\frac{n \cdot \log(n)}{p} + \log(n)\right) + O\left(\frac{n}{p} + \log(p)\right) + O\left(\frac{n}{p} + \log(p)\right) = O\left(\frac{n \cdot \log(n)}{p} + \log(n) + \log(p)\right).$$

## XIX. Example

To illustrate how this parallelization works, let us again use Stage 3. Specifically, let us show how we can use $p$ processors to compute the maximum of given $n+1$ values $V^{(0)}, \ldots, V^{(n)}$ in parallel in time $O\left(\frac{n}{p} + \log(p)\right)$.

Indeed, let us divide $n+1$ values into $p$ subgroups with $\frac{n+1}{p}$ elements in each subgroup. To each of these subgroups, we assign one of the $p$ processors. Each processor computes the maximum of all its $\frac{n+1}{p}$ values in time $\frac{n+1}{p} = O\left(\frac{n}{p}\right)$, and these processors work in parallel. After that, we have $p$ values – the maximum of the first subgroup, the maximum of the second subgroup, etc.

To find the maximum of all $n+1$ elements, it is now sufficient to find the largest of these $p$ subgroup maxima. We already know that if we have $p$ processors, then we can compute the maximum of $p$ values in parallel in time $O(\log(p))$.

Thus, we have a two-step process for computing the maximum. The first step requires time $O\left(\frac{n}{p}\right)$, the second step requires time $O(\log(p))$. Thus, the total computation time of this two-step process is indeed equal to $O\left(\frac{n}{p} + \log(p)\right)$.

## References


[1] G. E. Blelloch, "Prefix sums and their applications", In: J. H. Reif, *Synthesis of Parallel Algorithms*, Morgan Kaufmann, San Mateo, California, 1993, pp. 35–60.

[2] P. van der Broek and J. Noppen, "Fuzzy weighted average: alternative approach", *Proceedings of the 25th International Conference of the North American Fuzzy Information Processing Society NAFIPS'2006*, Montreal, Quebec, Canada, June 3–6, 2006.

[3] Th. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 2001.

[4] E. Dantsin, V. Kreinovich, A. Wolpert, and G. Xiang, "Population Variance under Interval Uncertainty: A New Algorithm", *Reliable Computing*, vol. 12, no. 4, pp. 273–280, 2006.

[5] D. Dubois, H. Fargier, and J. Fortin, "The empirical variance of a set of fuzzy intervals", *Proceedings of the 2005 IEEE International Conference on Fuzzy Systems FUZZ-IEEE'2005*, Reno, Nevada, May 22–25, 2005, pp. 885–890.

[6] S. Ferson, V. Kreinovich, J. Hajagos, W. Oberkampf, and L. Ginzburg, *Experimental Uncertainty Estimation and Statistics for Data Having Interval Uncertainty*, Sandia National Laboratories, Report SAND2007-0939, May 2007.

[7] P. Hansen, M. V. P. de Aragao, and C. C. Ribeiro, "Hyperbolic 0-1 programming and optimization in information retrieval", *Math. Programming*, vol. 52, pp. 255–263, 1991.

[8] J. Jájá, *An Introduction to Parallel Algorithms*, Addison-Wesley, Reading, MA, 1992.

[9] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic: theory and applications.* Prentice Hall, Upper Saddle River, New Jersey, 1995.

[10] V. Kreinovich, L. Longpré, S. A. Starks, G. Xiang, J. Beck, R. Kandathi, A. Nayak, S. Ferson, and J. Hajagos, "Interval Versions of Statistical Techniques, with Applications to Environmental Analysis, Bioinformatics, and Privacy in Statistical Databases", *Journal of Computational and Applied Mathematics*, vol. 199, No. 2, pp. 418–423, 2007.

[11] V. Kreinovich, G. Xiang, S. A. Starks, L. Longpre, M. Ceberio, R. Araiza, J. Beck, R. Kandathi, A. Nayak, R. Torres, and J. Hajagos, "Towards combining probabilistic and interval uncertainty in engineering calculations: algorithms for computing statistics under interval uncertainty, and their computational complexity", *Reliable Computing*, vol. 12, no. 6, pp. 471–501, 2006.

[12] H. T. Nguyen and E. A. Walker, *A first course in fuzzy logic*, CRC Press, Boca Raton, Florida, 2005.

[13] S. Rabinovich, *Measurement Errors and Uncertainties: Theory and Practice*, Springer-Verlag, New York, 2005.

[14] S. A. Vavasis, *Nonlinear Optimization: Complexity Issues*, Oxford University Press, New York, 1991.

[15] G. Xiang, M. Ceberio, and V. Kreinovich, "Computing Population Variance and Entropy under Interval Uncertainty: Linear-Time Algorithms", *Reliable Computing*, vol. 13, no. 6, pp. 467–488, 2007.