Towards Discrete Interval, Set, and Fuzzy Computations

Enrique Portillo  
*University of Texas at El Paso*, eportillo2@miners.utep.edu

Olga Kosheleva  
*University of Texas at El Paso*, olgak@utep.edu

Vladik Kreinovich  
*University of Texas at El Paso*, vladik@utep.edu

Follow this and additional works at: [http://digitalcommons.utep.edu/cs_techrep](http://digitalcommons.utep.edu/cs_techrep)

Part of the Computer Sciences Commons

Comments:
Technical Report: UTEP-CS-13-23a  

Recommended Citation
[http://digitalcommons.utep.edu/cs_techrep/753](http://digitalcommons.utep.edu/cs_techrep/753)

This Article is brought to you for free and open access by the Department of Computer Science at DigitalCommons@UTEP. It has been accepted for inclusion in Departmental Technical Reports (CS) by an authorized administrator of DigitalCommons@UTEP. For more information, please contact lweber@utep.edu.
Towards Discrete Interval, Set, and Fuzzy Computations

Enrique Portillo\textsuperscript{1}, Olga Kosheleva\textsuperscript{2}, and Vladik Kreinovich\textsuperscript{1}

Departments of \textsuperscript{1}Computer Science and \textsuperscript{2}Teacher Education
University of Texas at El Paso, El Paso, Texas 79968, USA
eportillo2@miners.utep.edu, olgak@utep.edu, vladik@utep.edu

Abstract—In many applications, we know the function \( f(x_1, \ldots, x_n) \), we know the intervals \( x_i \) of possible values of each quantity \( x_i \), and we are interested in the range of possible values of \( y = f(x_1, \ldots, x_n) \); this problem is known as the problem of interval computations. In other applications, we know the function \( f(x_1, \ldots, x_n) \), we know the fuzzy sets \( X_i \) that describe what we know about each quantity \( x_i \), and we are interested in finding the fuzzy set \( Y \) corresponding to the quantity \( y = f(x_1, \ldots, x_n) \); this problem is known as the problem of fuzzy computations. There are many efficient algorithms for solving these problems; however, most of these algorithms implicitly assume that each quantity \( x_i \) can take any real value within its range. In practice, some quantities are discrete: e.g., \( x_i \) can describe the number of people. In this paper, we provide feasible algorithms for interval, set, and fuzzy computations for such discrete inputs.

Index Terms—discrete case, interval computations, fuzzy computations

I. NEED FOR DISCRETE INTERVAL, SET, AND FUZZY COMPUTATIONS

Need for data processing. We want to understand the world, we want to know the values of different quantities which characterize the world. Some quantities we can directly measure: e.g., we can easily measure the temperature in El Paso. In many real-life situations, however, we are interested in the value of some quantity \( y \) which is difficult (or even impossible) to measure directly. For example, we may be interested in the temperature inside a star, or we may want to know where a close-to-Earth asteroid will be in 20 years.

Since we cannot measure these quantities directly, what we can do instead is:

- measure (or otherwise estimate) the values of the quantities \( x_1, \ldots, x_n \) which are related to \( y \) by a known relation \( y = f(x_1, \ldots, x_n) \), and then
- apply the corresponding algorithm \( f \) to the measurement results \( \bar{x}_1, \ldots, \bar{x}_n \), producing an estimate

\[
\bar{y} = f(\bar{x}_1, \ldots, \bar{x}_n)
\]

for \( y \).

For example, to measure the density at different depths and different locations, we can:

- measure the gravity field at different locations and at different heights, and then
- use the known algorithms to reconstruct the desired density values.

In the general case, such a measure-and-compute procedure is called indirect measurement, and the application of the algorithm \( f \) is known as data processing; see, e.g., [13].

Need to take uncertainty into account. Measurements are never absolutely accurate — and alternatives like expert estimates are even less accurate. The result \( \bar{x}_i \) of measuring (or estimating) the quantity \( x_i \) is, in general, different from the actual value of this quantity. Because of the corresponding approximation errors \( \Delta x_i \equiv \bar{x}_i - x_i \), the result \( \bar{y} = f(\bar{x}_1, \ldots, \bar{x}_n) \) of data processing is, in general, different from the actual (unknown) value \( y = f(x_1, \ldots, x_n) \) (even if we, for simplicity, assume that the dependence \( f \) between \( x_i \) and \( y \) is known exactly).

To make appropriate decisions based on the estimate \( \bar{y} \), it is important to know the accuracy \( \Delta y \equiv \bar{y} - y \) of this estimate. For example, if a geophysical analysis has shown that a natural gas field contains \( \bar{y} = 10 \) trillion cubic feet of gas, the big question is how accurate is this estimate:

- if it is \( 10 \pm 1 \), we should start production;
- if it is \( 10 \pm 20 \), meaning that there may be no gas at all, then a further geophysical analysis is needed before we invest a lot of money in this field.

Traditional probabilistic approach to uncertainty. Traditionally, in science and engineering, a probabilistic approach is used to gauge to measurement uncertainty. In this approach, we:

- estimate the probabilities of different values of \( \Delta x_i \) (it is often Gaussian),
- find correlations (if any) between the corresponding random variables \( \Delta x_i \), and then
- use known statistical methods to derive the resulting probability distribution for \( \Delta y \);

see, e.g., [13], [14].

A usual way of finding the probability distribution for \( \Delta x_i = \bar{x}_i - x_i \) is to repeatedly compare:

- the results \( \bar{x}_i \) obtained by our measuring instrument and
- the results \( \bar{x}_i^{\text{est}} \) obtained by a much more accurate (“standard”) measuring instrument.

Since the standard measuring instrument is much more accurate, we can ignore the corresponding measurement error \( \Delta x_i^{\text{est}} \) in comparison with \( \Delta x_i \) and thus, assume that the value \( \bar{x}_i^{\text{est}} \) measured by the standard instrument is equal to \( x_i \). Under this
assumption, the difference \( \bar{x}_i - \bar{x}_{i}^\dagger \) is approximately equal to the desired measurement uncertainty \( \Delta x_i = \bar{x}_i - x_i \). Based on these differences \( \bar{x}_i - \bar{x}_{i}^\dagger \approx \Delta x_i \), we can estimate the probability distribution for \( \Delta x_i \).

**Need to go beyond the traditional probabilistic approach.** There are two cases when the above approach cannot be applied.

- The first case if when we have state-of-the-art measurements. For example, a geophysicist may use the latest state-of-the-art super-accurate instrument for measuring the gravity values at different points. It would be great if there was a five times more accurate instrument available, but this instrument is the best we have.

- Another case is when we use the measurements as a part of manufacturing process. In such situations, in principle, we can calibrate each sensor, but the problem is that while sensors are often cheap, their calibration is several order of magnitude more expensive. Such an expensive calibration may be necessary if we are manufacturing a critical component of a passenger airplane or of a reactor for a nuclear power station, but when we manufacture toothbrushes, such expenses are unnecessary.

**Case of interval uncertainty.** When we do not know the probabilities of different values of measurement error \( \Delta x_i \), then we should know at least some upper bound \( \Delta_i \) on this error. Indeed, if we do not even known any upper bound, then this is not a measurement, this is a wild guess.

For measurements, a manufacturer provides an upper bound on the measurement error, i.e., a value \( \Delta_i \) for which

\[
|\Delta x_i| \leq \Delta_i.
\]

In this case, if we know the measurement result \( \bar{x}_i \), we can conclude that the actual value \( x_i \) of the measured quantity lies in the interval \( x_i = [\bar{x}_i - \Delta_i, \bar{x}_i + \Delta_i] \).

For example, when the measured value is \( \bar{x}_i = 1.0 \), and the accuracy is \( \Delta_i = 0.1 \), we conclude that the actual value \( x_i \) belongs to the interval \([1.0 - 0.1, 1.0 + 0.1] = [0.9, 1.1] \).

**Need for interval computations.** When we know all the inputs with interval uncertainty, then for each input \( x_i \), we only know the interval \( x_i \) of possible values of \( x_i \). Different combinations of values \( x_i \) from the corresponding intervals lead, in general, to different values \( y = f(x_1, \ldots, x_n) \) of the desired quantity \( y \). In such situations, it is desirable to find the set of all possible values \( y \), i.e., the set

\[
Y \overset{\text{def}}{=} \{ f(x_1, \ldots, x_n) : x_1 \in X_1, \ldots, x_n \in X_n \}.
\] (1)

The problem of estimating such a range based on known intervals \( X_i \) is known as a problem of interval computations; see, e.g., [6], [9].

**Comment.** In a more general case, when instead of intervals, we have more general sets, we get what is called set computations.

**Need for fuzzy uncertainty and fuzzy computations.** In many practical cases, instead of measuring the values \( x_i \), we ask experts to estimate these values. Such estimates are ubiquitous in such difficult-to-formalize areas as medicine, and even in such easier-to-formalize areas as geophysics. Experts can rarely express their estimates by using exact numbers, their estimates usually involve imprecise (“fuzzy”) words from a natural language. For example, an expert can say that the value is small, or, in more detail, that is approximately equal to 1.0, with an error of approximately 0.1.

Fuzzy logic is a technique specifically invented to translate such imprecise statements into a language understandable to a computer; see, e.g., [8], [11], [15]. In fuzzy logic, to each statement \( S \) describing a real number, and to each possible value \( x \), we assign a degree \( \mu_S(x) \) describing to what extent the value \( x \) is consistent with this statement. This degree can be obtained, e.g., by asking an expert to mark his/her confidence that \( x \) is consistent with \( S \) on a scale from 0 to some integer \( n \); if an expert marks \( m \), then we take

\[
\mu_S(x) = \frac{m}{n}.
\]

This procedure can be repeated for several values \( x \), and then an interpolation can be used to find the values \( \mu_S(x) \) for all other real values. The corresponding function \( \mu_S(x) \) is known as a membership function.

Once we have membership functions \( \mu_i(x_i) \) corresponding to different inputs, we need to compute the membership function \( \mu(y) \) corresponding to \( y = f(x_1, \ldots, x_n) \). The computation of such membership function is known as fuzzy computation.

**How to perform fuzzy computations: towards Zadeh’s extension principle.** In order to describe formulas for fuzzy computations, it is reasonable to recall that \( y \) is a possible value of the desired variable if for some real numbers \( x_1, \ldots, x_n \) for which \( y = f(x_1, \ldots, x_n) \):

- the value \( x_1 \) is a possible value of the first input, and
- the value \( x_2 \) is a possible value of the second input, and
- \( \ldots \)
- the value \( x_n \) is a possible value of the \( n \)-th input.

In other words, we are interested in the degree to which the following statement holds:

\[
\bigvee_{x_i : f(x_1, \ldots, x_n) = y} (x_1 \text{ is possible } \& \ldots \& x_n \text{ is possible}). \tag{2}
\]

We know the degree to which each \( x_i \) is a possible value of the \( i \)-th input; this degree is equal to \( \mu_i(x_i) \). The highlighted statement is obtained from these basic statements by using “or” \((\lor)\) and “and” \((\&)\). Thus, to find the degree in the statement (2), we need to be able to transform the degrees of belief \( a \) and \( b \) in original \( A \) and \( B \) into degrees of belief into their propositional combinations \( A \lor B \) and \( A \& B \).

This problem is typical in the analysis of expert statements. Indeed, the degrees to which \( A \& B \) and \( A \lor B \) are true depend not only on the degrees with which the statements \( A \) and \( B \) are true, but also on the relations between \( A \) and \( B \). So,
ideally, after we ask the experts about their degrees of certainty in different statements $S_1, \ldots, S_n$, we should also ask them about their degrees of certainty in different propositional combinations of these statements. However, there are exponentially many such combinations, and it is not practically possible to elicit all corresponding degrees. Thus, we have to able, given degrees $a$ and $b$ for statements $A$ and $B$, generate reasonable estimates $f_v(a, b)$ for the degree of $A \lor B$ and $f_k(a, b)$ for the degree of $A \land B$.

It is reasonable to require that these estimates satisfy reasonable properties: e.g., since “$A$ and true” and “$A$ and $A$” are both equivalent to $A$, we should have $f_k(a, 1) = f_k(a, a) = a$. The more we believe in $B$, the larger is our belief in $A \land B$, so we should have $b \leq b'$ imply $f_k(a, b) \leq f_k(a, b')$. Thus, when $a \leq b \leq 1$, we have

$$a = f_k(a, a) \leq f_k(a, b) \leq f_k(a, 1) = a$$

hence $f_k(a, b) = a$. Since “$A$ or true” and “$A$ or $A$” are both equivalent to $A$, we should have $f_v(a, 0) = f_v(a, a) = a$. The more we believe in $B$, the larger is our belief in $A \lor B$, so we should have $b \leq b'$ imply $f_v(a, b) \leq f_v(a, b')$. Thus, when $0 \leq b \leq a$, we have

$$a = f_v(a, 0) \leq f_v(a, b) \leq f_v(a, a) = a$$

hence $f_v(a, b) = a$. Since “$A$ or $B$” means the same as “$B$ or $A$”, we have $f_v(b, a) = f_v(a, b) = a$, i.e., in general,

$$f_v(a, b) = (\min(a, b)).$$

Substituting $\min$ and $\max$ into the formula (2), we conclude that

$$\mu(y) = \max_{x_1, \ldots, x_n/f(x_1, \ldots, x_n)} \min(\mu_1(x_1), \ldots, \mu_n(x_n)).$$  \hspace{1cm} (3)

This formula, first proposed by L. Zadeh, the founder of fuzzy logic, is known as Zadeh’s extension principle.

From the computational viewpoint, fuzzy computations can be reduced to interval and set computations. According to the formula (3), for every real number $\alpha$, the inequality

$$\mu(y) \geq \alpha$$

is equivalent to the condition that for some $x_1, \ldots, x_n$ for which $f(x_1, \ldots, x_n) = y$, we have $\mu_1(x_1) \geq \alpha$, $\ldots$, and $\mu_n(x_n) \geq \alpha$. In other words, if we denote

$$x_1(\alpha) \triangleq \{x_1 : \mu_1(x_1) \geq \alpha\}$$

and $y(\alpha) = \{y : \mu(y) \geq \alpha\}$, we get

$$y(\alpha) = \{f(x_1, \ldots, x_n) : x_1 \in x_1(\alpha), \ldots, x_n(\alpha)\}. \hspace{1cm} (4)$$

By applying interval (or set) computations, we can find the sets $y(\alpha) = \{y : \mu(y) \geq \alpha\}$. Once we know these sets, we can reconstruct each value $\mu(y)$ of the desired membership function $\mu$ as $\max\{\alpha : \mu(y) \geq \alpha\}$, i.e., as $\max\{\alpha : y \in y(\alpha)\}$.

In other words, fuzzy computations can be reduced to the interval (or, more generally, set) computations for the sets $x_i(\alpha)$ (such sets are known as $\alpha$-cuts).

Need for discrete computations. Usually, in interval, set and fuzzy computations, we consider the cases when all the variables are continuous, i.e., can take all real values from the corresponding ranges. In practice, sometimes, variables are discrete, e.g., $x_i$ can be number of people. It is reasonable to develop efficient algorithms for computing the ranges in such discrete cases.

Comment. In view of the above reduction of fuzzy computations to interval (set) computations, in the following text, we will consider only algorithms for the interval (set) case.

II. IN GENERAL, THE CORRESPONDING DISCRETE INTERVAL AND SET COMPUTATION PROBLEMS ARE COMPUTATIONALLY INTRACTABLE (NP-HARD)

How difficult are the usual (continuous) interval computation problems. For a linear function

$$f(x_1, \ldots, x_n) = a_0 + \sum_{i=1}^{n} a_i \cdot x_i, \hspace{1cm} (5)$$

its range $y$ over intervals $x_i = [\bar{x}_i - \Delta_i, \bar{x}_i + \Delta_i]$ is easy to compute: $y = [\bar{y} - \Delta, \bar{y} + \Delta]$, where $\bar{y} = a_0 + \sum_{i=1}^{n} a_i \cdot \bar{x}_i$ and

$$\Delta = \sum_{i=1}^{n} |a_i| \cdot \Delta_i.$$  

For a quadratic function

$$f(x_1, \ldots, x_n) = a_0 + \sum_{i=1}^{n} a_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \cdot x_i \cdot x_j,$$

computing the range over given intervals is, in general, an NP-hard problem; see, e.g., [7] and references therein. Moreover, it is NP-hard even when we restrict ourselves to such a simple quadratic function as variance [3], [4], [10]:

$$V = \frac{1}{n} \sum_{i=1}^{n} x_i^2 - \left(\frac{1}{n} \sum_{i=1}^{n} x_i\right)^2. \hspace{1cm} (6)$$

Comment. This comment is for those who are not very familiar with the notion of NP-hardness. This notion is related to the notion of a feasible algorithm:

- some algorithms are feasible in the sense that they finish in reasonable time,
- some algorithms are not feasible, since they take time – even for reasonable-size inputs – which is longer than the lifetime of the Universe.

It is usually assumed that feasible algorithms are exactly polynomial-time algorithms, i.e., algorithms $A$ for which the running time $t_A(x)$ on each input $x$ is bounded by a polynomial of the bit-length $\text{len}(x)$ of the input: $t_A(x) \leq P(\text{len}(x))$. This definition is not perfect:
• an algorithm which requires time $10^{300} \cdot \text{len}(x)$ is clearly polynomial-time, but not feasible;
• on the other hand, an algorithm that finishes in time $\exp\left(10^{-7} \cdot \text{len}(x)\right)$ is clearly not polynomial-time, but it is also clearly feasible for all inputs $x$ of length $\text{len}(x) \leq 10^3$.

However, while not perfect, this is the best definition we have.

We then consider a class of problems (called NP) in which:
• once we have a guess,
• we can check, in feasible time, whether this guess is a correct solution.

Some problems from the class NP can be solved in polynomial time. For other problems, no such feasible algorithm is known. Whether every problem from the class NP can be solved in feasible time is an open problem; this problem is known as $P=\text{NP}$ (Most computer scientists believe that $P \neq \text{NP}$). What is known is that some problem from the class NP are the hardest in this class, in the sense that every problem from the class NP can be feasibly reduced to this problem. Such problems are known as NP-hard. (For more detailed explanations, see [5], [7], [12].)

**Discrete case.** In the discrete case, each variable $x_i$ can take only finitely many values $x_{i1}, \ldots, x_{i\alpha_i}$. In practical situations, these values are equally distributed, i.e., have the form $x_{ij} = c_i + j \cdot h_i$ for some starting point $c_i$ and step $h_i$. In other words, the value $x_i$ can be expressed as $x_i = c_i + x'_i \cdot h_i$, where the new variable $x'_i$ can only take integer values. We can substitute such expressions for $x_i$ into the desired dependence $y = f(x_1, \ldots, x_n)$ and thus, get a new expression in which each new variable takes only integer values. Without losing generality, we can therefore assume that each variable $x_i$ takes integer values between some bounds $\underline{X}_i$ and $\overline{X}_i$. In this case, the range estimation problem takes the following form:
• for each input $i$, we know the bounds $\underline{X}_i$ and $\overline{X}_i$ on $x_i$;
• we also know a function $f(x_1, \ldots, x_n)$;
• our objective is to find the range
  $$Y = \{f(x_1, \ldots, x_n) : x_i = \underline{X}_i, \overline{X}_i + 1, \ldots, \overline{X}_i\}.$$  

**Discrete case: the problem becomes NP-hard even for linear functions.** When the values $x_i$ are discrete, then even for interval functions $f(x_1, \ldots, x_n)$, the problem of computing the range becomes NP-hard. This directly follows from the known fact that the following subset sum problem is NP-hard [5], [7], [12]:
• given integers $a_1, \ldots, a_n$ and $a$,
• check whether there exist values $x_i \in \{0, 1\}$ for which
  $$\sum_{i=1}^n a_i \cdot x_i = a.$$  
This means that when we consider variables $x_i$ that take only values $0$ and $1$, it is NP-hard to check whether the range of a linear function $f(x_1, \ldots, x_n) = \sum_{i=1}^n a_i \cdot x_i$ contains given integer $a$.

**What we do in this paper.** In this paper, we use the ideas originally proposed by N. Vereschagin and E. Hirsh in their unpublished work and ideas from [1], [2] to show that in a realistic situation when the ranges of the coefficients $a_i$ are bounded, the problem of estimating the range of a linear function for discrete inputs becomes feasible. Moreover, we show that the problem of computing the range is feasible even for some quadratic functions such as variance.

III. EFFICIENT ALGORITHMS FOR DISCRETE INTERVAL (AND SET) COMPUTATIONS

**Main assumption.** In practice, there is usually a general bound on the values of all the coefficients $a_i$. In line with this fact, let us assume that all the values are bounded by some constant $A$: $|a_i| \leq A$.

It is also reasonable to assume that the possible values of $x_i$ are bounded by some constant $X$: $|x_i| \leq X$. In other words, we have $|\underline{X}_i| \leq X$ and $|\overline{X}_i| \leq X$ for all $i$.

In this section, we show that under these reasonable assumptions, we can design efficient algorithms for computing the range of a linear function (and even of some quadratic functions).

**Case of a linear function: at first glance, it looks like we need exponential time.** For a linear function (5), we want to find the set of all its values when each inputs takes all the values $x_i$ from $\underline{X}_i$ to $\overline{X}_i$. There are finitely many values of each input $x_i$, so there are finitely many combinations $(x_1, \ldots, x_n)$ of such values. In principle, we can:
• enumerate all such combinations,
• compute the value $f(x_1, \ldots, x_n)$ for each of these combinations, and
• thus form the desired set $Y$.

However, if we take at least two different values of each of $n$ variables $x_i$, we will thus need to consider at least $2^n$ different combinations $(x_1, \ldots, x_n)$. Thus, the above straightforward algorithm requires exponential time.

**Case of a linear function: a new polynomial-time algorithm.** To decrease the computation time, instead of directly computing the desired range
  $$Y = \left\{ \sum_{i=1}^n a_i \cdot x_i : x_i = \underline{X}_i, \overline{X}_i + 1, \ldots, \overline{X}_i \right\},$$  
let us sequentially compute the ranges $Y_0 = \{0\}$, $Y_1$, $Y_2$, $\ldots$, $Y_k$, $\ldots$, $Y_{n-1}$, and then finally $Y_n = Y$, where we denoted
  $$Y_k = \left\{ \sum_{i=1}^k a_i \cdot x_i : x_i = \underline{X}_i, \overline{X}_i + 1, \ldots, \overline{X}_i \right\}.$$  

For each $k$, from $|a_i| \leq A$ and $|x_i| \leq X$, we conclude that $|a_i \cdot x_i| \leq A \cdot X$, and thus, $\sum_{i=1}^k |a_i \cdot x_i| \leq k \cdot A \cdot X \leq n \cdot A \cdot X$. So, each set $Y_k$ only contains integers from $-n \cdot A \cdot X$ to $n \cdot A \cdot X$. There are no more than $2n \cdot A \cdot X + 1$ such integers, so each set $Y_k$ contains no more than $2n \cdot A \cdot X + 1$ elements.

Once the set $Y_k$ is computed, we can compute $Y_{k+1}$ by using the fact that

$$Y_{k+1} = \left\{ \sum_{i=1}^{k+1} a_i \cdot x_i : x_i = \underline{X}_i, \overline{X}_i + 1, \ldots, \overline{X}_i \right\}.$$
\[ Y_{k+1} = \{ y_k + a_{k+1} \cdot x_{k+1} : y_k \in Y_k \} \]

\[ x_{k+1} = \sum_{i=1}^{k+1} x_i. \]

We can compute this set by taking, for each elements \( y_k \) from the set \( Y_k \), all possible values \( x_{k+1} \), and computing the values \( y_k + a_{k+1} \cdot x_{k+1} \). The set \( Y_k \) contains no more than

\[ 2n \cdot A \cdot X + 1 = O(n) \]

elements, and there are at most \( 2X + 1 = O(1) \) possible values of \( x_{k+1} \); so overall, we need to consider \( O(n) \cdot O(1) = O(n) \) pairs.

After \( n \) iterations of this procedure, we get the desired range \( Y = Y_n \). In this computation, we repeat \( O(n) \) steps \( n \) times. So, the overall computation time is equal to \( n \cdot O(n) = O(n^2) \). This algorithm requires quadratic time.

**Comment.** Please note that this algorithm (and following algorithms) compute the exact range in polynomial time. In other words, we gain computation time without sacrificing accuracy.

**Computing the range of variance in polynomial time.** To compute the variance, we need to know the values \( y = \sum_{i=1}^{n} x_i^2 \) and \( z = \sum_{i=1}^{n} x_i \). To compute the corresponding ranges in polynomial time, similarly to the case of the linear function, let us sequentially compute the intermediate sets of pairs \( P_0 = \{ (0,0) \}, P_1, \ldots, P_n \), where

\[ P_k = \left\{ \left( \sum_{i=1}^{k} x_i^2, \sum_{i=1}^{k} x_i \right) : x_i = X_1, \ldots, X_k \right\}. \]

For each \( k \), there are \( O(n) \) possible values of \( \sum_{i=1}^{k} x_i^2 \) and \( O(n) \) possible values of \( \sum_{i=1}^{k} x_i \). So, for each \( k \), the set \( P_k \) contains no more than \( O(n) \cdot O(n) = O(n^2) \) possible pairs. Once we know \( P_k \), we can find \( P_{k+1} \) as

\[ P_{k+1} = \{ (y_k + x_{k+1}^2, z_k + x_{k+1}) : (y_k, z_k) \in P_k \} \]

To compute \( P_{k+1} \), for each of \( O(n^2) \) possible pairs \( (y_k, z_k) \in P_k \), we consider all \( O(1) \) possible values \( x_{k+1} \); thus, each computation requires \( O(n^2) \) computations steps.

After \( n \) iterations, we get the set \( P = P_n \). These computations take time \( n \cdot O(n^2) = O(n^3) \).

Once we have the set \( P_n \), we can compute the desired set \( Y \) of possible values of variance as

\[ Y = \left\{ \frac{1}{n} \cdot y - \left( \frac{1}{n} \cdot z \right)^2 : (y, z) \in P_n \right\}. \]

For each of \( O(n^2) \) pairs \( (y, z) \in P \), computing the expression

\[ \frac{1}{n} \cdot y - \left( \frac{1}{n} \cdot z \right)^2 \]

takes \( O(1) \) steps. Thus, the overall computation of \( Y \) from \( P \) takes \( O(n^2) \cdot O(1) = O(n^2) \) steps.

The overall computation time for this algorithm is therefore equal to \( O(n^3) + O(n^2) = O(n^3) \).

**Computing higher central moments in polynomial time.** The above construction can be generalized to computing higher-order central moments. Indeed, for each positive integer \( m \), the \( m \)-th central moment is defined as

\[ C_m = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^k, \]

where \( E = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \). By using the usual binomial formula to describe \( (x_i - E)^k \), we can represent \( C_m \) as a linear combination of the sums

\[ M_{\ell} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^{\ell}, \]

for \( \ell = 0, 1, 2, \ldots, n \), with coefficients proportional to different powers of \( E = M_1 \).

To compute the range of possible values of \( C_m \), let us therefore compute, for each \( k \) from 0 to \( n \), the set \( T_k \) of all possible tuples \( (s_1(k), \ldots, s_m(k)) \), where \( s_k(k) = \sum_{i=1}^{n} x_i^{k} \). For \( k = 0 \), we have \( T_0 = \{(0, \ldots, 0)\} \). Each sum \( s_k \) is bounded by \( O(n) \), so for each \( k \), the set \( T_k \) contains no more than \( O(n^m) \) tuples.

Once we know the set \( T_k \), we can compute \( T_{k+1} \) as

\[ T_{k+1} = \{ (s_1 + x_{k+1}, \ldots, s_m + x_{k+1}^m) : (s_1, \ldots, s_m) \in T_k \} \]

On each step, this computation takes time \( O(n^m) \), so overall, the computation of the set \( T_n \) takes time

\[ n \cdot O(n^m) = O(n^{m+1}). \]

Once we have the set \( T_n \), we can take each of its \( O(n^m) \) tuples, and for each of them, compute the desired moment \( M_n \). This takes time \( O(n^{m+1}) \), so overall, this algorithm takes time \( O(n^{m+1}) + O(n^m) = O(n^{m+1}) \).

**Computing covariance in polynomial time.** For covariance

\[ C_{xy} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \cdot y_i - \left( \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \right) \cdot \left( \frac{1}{n} \cdot \sum_{i=1}^{n} y_i \right), \]

we can similarly sequentially compute the sets of triples \( T_0 = \{(0, 0, 0)\}, T_1, \ldots, T_n \), where

\[ T_k = \left\{ \left( \sum_{i=1}^{k} x_i^2, \sum_{i=1}^{k} y_i, \sum_{i=1}^{k} x_i \cdot y_i \right) \right\}. \]

Each component has \( O(n) \) values, so each set consists of \( O(n^3) \) elements. Here,

\[ T_{k+1} = \{ (s_1 + x_{k+1}, s_2 + y_{k+1}, s_3 + x_{k+1} \cdot y_{k+1}) : (s_1, s_2, s_3) \in T_k \} \]

For each set \( T_k \), we can compute the desired set \( Y \) of possible values of variance as

\[ Y = \left\{ \frac{1}{n} \cdot y - \left( \frac{1}{n} \cdot z \right)^2 : (y, z) \in P_n \right\}. \]
Each iteration thus takes $O(n^3)$ steps, so the overall computation time for $T_n$ is $n \cdot O(n^3) = O(n^4)$.

Once we know the set $T_n$ with $O(n^3)$ triples $(s_1, s_2, s_3)$, we can compute the covariance $C_{xy}$ for each of these triples, and thus, find the desired range $Y$ of the covariance in time $O(n^3)$.

The overall computation time is thus $O(n^4) + O(n^3) = O(n^4)$.

**Computing correlation in polynomial time.** A similar idea can be used to compute the range of correlation in polynomial time. Indeed, the correlation $\rho_{xy}$ is defined as

$$ \rho_{xy} = \frac{C_{xy}}{\sqrt{V_x V_y}}, $$

where

$$ V_x \equiv \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2 - \left( \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \right)^2 $$

and

$$ V_y \equiv \frac{1}{n} \cdot \sum_{i=1}^{n} y_i^2 - \left( \frac{1}{n} \cdot \sum_{i=1}^{n} y_i \right)^2. $$

To compute the range of the correlation, we compute the sets of tuples $T_0 = \{(0, 0, 0, 0, 0)\}$, $T_1, \ldots, T_n$, where

$$ T_k = \left\{ \left( \sum_{i=1}^{k} x_i, \sum_{i=1}^{k} y_i, \sum_{i=1}^{k} x_i y_i, \sum_{i=1}^{k} x_i^2, \sum_{i=1}^{k} y_i^2 \right) \right\}. $$

Each component has $O(n)$ values, so each set consists of $O(n^5)$ elements. Here,

$$ T_{k+1} = \left\{ (s_1 + x_{k+1}, s_2 + y_{k+1}, s_3 + x_{k+1} y_{k+1}, s_4 + x_{k+1}^2, s_5 + y_{k+1}^2) : (s_1, s_2, s_3, s_4, s_5) \in T_k, \& \right. $$

$$ x_{k+1} = \sum_{i=1}^{k} x_i, \left. \& \right. \sum_{i=1}^{k} y_i $$$$ y_{k+1} = \sum_{i=1}^{k} x_i \cdot y_i, \sum_{i=1}^{k} x_i^2, \sum_{i=1}^{k} y_i^2 \right\}. $$

Each iteration thus takes $O(n^5)$ steps, so the overall computation time for $T_n$ is $n \cdot O(n^5) = O(n^6)$.

Once we know the set $T_n$ with $O(n^3)$ tuples $(s_1, s_2, s_3)$, we can compute $C_{xy}$, $V_x$, $V_y$, and the correlation $\rho_{xy}$ for each of these tuples, and thus, find the desired range $Y$ of the covariance in time $O(n^3)$. The overall computation time is thus $O(n^6) + O(n^3) = O(n^6)$, i.e., polynomial.

**IV. CONCLUSIONS**

In many practical situations, we have some information about the quantities $x_1, \ldots, x_n$, and we are interested in the quantity $y$ which is related to $x_i$ by a known dependence $y = f(x_1, \ldots, x_n)$. For example, if we know the range $[x_i, x_i]$ of each of the quantities $x_i$, then we want to find the range of possible values of $y$. If our information about each $x_i$ is described by a fuzzy set $X_i$, then we want to know the resulting fuzzy set for $Y$.

There are many efficient algorithms for solving this problem, but these algorithms assume that all values from each interval $[x_i, x_i]$ are possible. In practice, we sometimes have an additional information, e.g., that the values $x_i$ are integers. This information limits the range of $y$.

In this paper, we provide feasible algorithms for computing such limited ranges for several reasonable functions $f(x_1, \ldots, x_n)$.

**ACKNOWLEDGMENT**

This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (CyberShARE Center of Excellence) and DUE-0926721, by Grants 1 T36 GM078000-01 and 1R43TR000173-01 from the National Institutes of Health, and by a grant on F-transforms from the Office of Naval Research.

The authors are thankful to the anonymous referees for valuable suggestions.

**REFERENCES**


