Computational Complexity and Feasibility of Data Processing and Interval Computations, With Extension to Cases When We Have Partial Information about Probabilities

Vladik Kreinovich and Luc Longpré
Department of Computer Science, University of Texas at El Paso
El Paso, TX 79968, USA, vladik@cs.utep.edu

Abstract

In many real-life situations, we are interested in the value of a physical quantity y that is difficult or impossible to measure directly. To estimate y, we find some easier-to-measure quantities $x_1, \ldots, x_n$ which are related to y by a known relation $y = f(x_1, \ldots, x_n)$. Measurements are never 100% accurate; hence, the measured values $\widetilde{x}_i$ are different from x_i, and the resulting estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ is different from the desired value $y = f(x_1, \ldots, x_n)$. How different?

Traditional engineering to error estimation in data processing assumes that we know the probabilities of different measurement error $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$. In many practical situations, we only know the upper bound Δ_i for this error; hence, after the measurement, the only information that we have about x_i is that it belongs to the interval $\mathbf{x}_i \stackrel{\text{def}}{=} [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. In this case, it is important to find the range $\mathbf{y}$ of all possible values of $y = f(x_1, \ldots, x_n)$ when $x_i \in \mathbf{x}_i$. We start the paper with a brief overview of the computational complexity of the corresponding *interval computation* problems.

We then discuss how this computational complexity changes when, in addition to the upper bounds Δ_i, we have some partial information about the probabilities of different values of Δx_i.

We also show how the use of quantum computing can speed up some computations related to interval and probabilistic uncertainty.

Most of the related problems turn out to be, in general, at least NP-hard. We end the paper with speculations on whether (and how) hypothetic physical devices can compute NP-hard problems faster than in exponential time.

1 Introduction: Data Processing–From Computing to Probabilities to Intervals

Why data processing? In many real-life situations, we are interested in the value of a physical quantity y that is difficult or impossible to measure directly. Examples of such quantities are the distance to a star and the amount of oil in a given well. Since we cannot measure y directly, a natural idea is to measure y *indirectly*. Specifically, we find some easier-to-measure quantities $x_1, \ldots, x_n$ which are related to y by a known relation $y = f(x_1, \ldots, x_n)$; this relation may be a simple functional transformation, or complex algorithm (e.g., for the amount of oil, numerical solution to an inverse problem). Then, to estimate y, we first measure the values of the quantities $x_1, \ldots, x_n$, and then we use the results $\widetilde{x}_1, \ldots, \widetilde{x}_n$ of these measurements to to compute an estimate $\widetilde{y}$ for y as $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

For example, to find the resistance R, we measure current I and voltage V, and then use the known relation $R = V/I$ to estimate resistance as $\widetilde{R} = \widetilde{V}/\widetilde{I}$.

Computing an estimate for y based on the results of direct measurements is called *data processing*; data processing is the main reason why computers were invented in the first place, and data processing is still one of the main uses of computers as number crunching devices.

Comment. In this paper, for simplicity, we consider the case when the relation between x_i and y is known exactly; in some practical situations, we only known an approximate relation between x_i and y.

Why interval computations? From computing to probabilities to intervals. Measurement are never 100% accurate, so in reality, the actual value x_i of i-th measured quantity can differ from the measurement result $\widetilde{x}_i$. Because of these *measurement errors* $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$, the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of data processing is, in general, different from the actual value $y = f(x_1, \ldots, x_n)$ of the desired quantity y [59].

It is desirable to describe the error $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y$ of the result of data processing. To do that, we must have some information about the errors of direct measurements.

What do we know about the errors Δx_i of direct measurements? First, the manufacturer of the measuring instrument must supply us with an upper bound Δ_i on the measurement error. If no such upper bound is supplied, this means that no accuracy is guaranteed, and the corresponding "measuring instrument" is practically useless. In this case, once we performed a measurement and got a measurement result $\widetilde{x}_i$, we know that the actual (unknown) value x_i of the measured quantity belongs to the interval $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$, where $\underline{x}_i = \widetilde{x}_i - \Delta_i$ and $\overline{x}_i = \widetilde{x}_i + \Delta_i$.

In many practical situations, we not only know the interval $[-\Delta_i, \Delta_i]$ of possible values of the measurement error; we also know the probability of different values Δx_i within this interval. This knowledge underlies the traditional engineering approach to estimating the error of indirect measurement, in which we assume that we know the probability distributions for measurement errors Δx_i.

In practice, we can determine the desired probabilities of different values of Δx_i 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 use, 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. 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 Δx_i; the only information we have is the upper bound on the measurement error.

In this case, after we performed a measurement and got a measurement result $\widetilde{x}_i$, the only information that we have about the actual value x_i of the measured quantity is that it belongs to the interval $\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. In such situations, the only information that we have about the (unknown) actual value of $y = f(x_1, \ldots, x_n)$ is that y belongs to the range $\mathbf{y} = [\underline{y}, \overline{y}]$ of the function f over the box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$:

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

The process of computing this interval range based on the input intervals $\mathbf{x}_i$ is called *interval computations*; see, e.g., [25, 26, 27, 46].

Interval computations techniques: brief reminder. Historically the first method for computing the enclosure for the range is the method which is sometimes called "straightforward" interval computations. This method is based on the fact that inside the computer, every algorithm consists of elementary operations (arithmetic operations, min, max, etc.). For each elementary operation $f(a, b)$, if we know the intervals $\mathbf{a}$ and $\mathbf{b}$ for a and b, we can compute the exact range $f(\mathbf{a}, \mathbf{b})$. The corresponding formulas form the so-called *interval arithmetic*. For example,

$$[\underline{a}, \overline{a}] + [\underline{b}, \overline{b}] = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]; \quad [\underline{a}, \overline{a}] - [\underline{b}, \overline{b}] = [\underline{a} - \overline{b}, \overline{a} - \underline{b}];$$

$$[\underline{a}, \overline{a}] \cdot [\underline{b}, \overline{b}] = [\min(\underline{a} \cdot \underline{b}, \underline{a} \cdot \overline{b}, \overline{a} \cdot \underline{b}, \overline{a} \cdot \overline{b}), \max(\underline{a} \cdot \underline{b}, \underline{a} \cdot \overline{b}, \overline{a} \cdot \underline{b}, \overline{a} \cdot \overline{b})].$$

In straightforward interval computations, we repeat the computations forming the program f step-by-step, replacing each operation with real numbers by the corresponding operation of interval arithmetic. It is known that, as a result, we get an enclosure $\mathbf{Y} \supseteq \mathbf{y}$ for the desired range.

In some cases, this enclosure is exact. In more complex cases (see examples below), the enclosure has excess width.

There exist more sophisticated techniques for producing a narrower enclosure, e.g., a centered form method. However, for each of these techniques, there are cases when we get an excess width. Reason: as shown in [33, 66], the problem of computing the exact range is known to be NP-hard even for polynomial functions $f(x_1, \ldots, x_n)$ (actually, even for quadratic functions f).

2 Computational Complexity of Interval Computations: Brief Reminder

What exactly problem are we solving? By the *basic problem of interval computations*, we mean the following problem:

GIVEN:

- n rational intervals $\mathbf{x}_i$ (i.e., intervals with rational endpoints), and
- a computable continuous function f that transforms n real numbers $x_1, \ldots, x_n$ into a real number $y = f(x_1, \ldots, x_n)$.

COMPUTE: the interval of possible values of y:

$$\mathbf{y} = [\underline{y}, \overline{y}] = f(\mathbf{x}_1, \ldots, \mathbf{x}_n) =$$

$$\{y \,|\, y = f(x_1, \ldots, x_n) \text{ for some } x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

Here, by a *computable function* $f(x_1, \ldots, x_n)$, we mean an algorithm that, for arbitrary rational numbers $x_1, \ldots, x_n$, and $\delta > 0$, computes a rational number that is δ-close to $f(x_1, \ldots, x_n)$.

By *computing the interval* $\mathbf{y} = [\underline{y}, \overline{y}]$, we mean *computing its endpoints* $\underline{y}$ and $\overline{y}$. If these endpoints are not rational numbers, then computing these endpoints means being able to compute them with any given rational accuracy $\varepsilon > 0$, i.e., computing the rational numbers $\tilde{\underline{y}}$ and $\tilde{\overline{y}}$ for which $|\tilde{\underline{y}} - \underline{y}| \le \varepsilon$ and $|\tilde{\overline{y}} - \overline{y}| \le \varepsilon$. Thus, we arrive at the following definition:

By the $\varepsilon-$*approximate basic problem of interval computations*, we mean the following problem:

GIVEN:

- n rational intervals $\mathbf{x}_i$, and
- a computable continuous function f that transforms n real numbers $x_1, \ldots, x_n$ into a real number $y = f(x_1, \ldots, x_n)$;
- a rational number $\varepsilon > 0$.

COMPUTE: rational numbers $\tilde{\underline{y}}$ and $\tilde{\overline{y}}$ that are ε-close to the range's endpoints, i.e., for which $|\tilde{\underline{y}} - \underline{y}| \le \varepsilon$ and $|\tilde{\overline{y}} - \overline{y}| \le \varepsilon$, where:

$$\mathbf{y} = [\underline{y}, \overline{y}] = f(\mathbf{x}_1, \ldots, \mathbf{x}_n) =$$

$$\{y \mid y = f(x_1, \ldots, x_n) \text{ for some } x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$$

How we represent integers and real numbers. Inside the computer, *integers* are usually represented in their binary form. Therefore, the input length of an integer can be naturally defined as the number of bits in its binary expansion.

Similarly, *binary-rational* numbers, i.e., numbers of the type $p/2^q$, are usually represented in their fixed-point binary form: e.g., $3/8_{10} = 0.011_2$ takes 4 bits, while $19/16_{10} = 1.0011_2$ requires 5 bits. So, e.g., if we say that "there is a polynomial-time algorithm that, for every binary-rational number ε, computes ...", we mean, in particular, that for the values $\varepsilon_n = 0.0\ldots01_2 = 2^{-n}$ of length n, the running time of this algorithm is bounded by a polynomial of n.

A general *rational* number m/n, where m and n are integers, can be naturally represented as a *pair* of integers m and n; therefore, we take the total length of the binary representations of m and n as the input length: e.g., $5/7_{10} = 101/111_2$ requires 6 bits to describe.

In most computers, there is no special *rational* data type, there is a type *real* which actually describes binary rational numbers. In most computers, there are two *different* representations of these "real" numbers: in addition to the above-described *fixed-point* real numbers, there are also *floating-point* real numbers, in which a binary rational number is represented as $m \cdot 2^e$, where m is a fixed-point real number (usually, with only zero before the binary point) and e is an integer.

In this paper, we describe the input length in terms of the *fixed point* representation. Most of our results about computational complexity and feasibility of data processing and interval computations, both positive (that some problems can be solved by feasible algorithms) and negative (that for some other problems no feasible algorithm is possible) are true for floating point numbers as well: e.g., since every fixed point number is at the same time a floating point number (with $e = 0$), *negative* results about fixed point inputs are automatically transformed into *negative* results about the floating point inputs.

Complexity in what sense? In this paper, we consider the regular Turing machine-type complexity.

For Turing machines, we have clearly defined steps, so we can define computational complexity $t_{\mathcal{U}}(x)$ of an algorithm $\mathcal{U}$ on an input x as the number of the corresponding steps. This is the *standard definition* of computational complexity in theory of computing.

Turing machine is a very primitive computer, on which a simple operation that is hardware supported on a normal computer can take a very long time. So, the Turing machine-based complexity is a rather poor estimate for the *actual* computation time of an algorithm on a real computer. If we use more sophisticated computer models, we can get much *better* estimates.

What makes Turing machine-based complexity *standard* and widely used is the fact that although the *actual* computation time changes from one type of computer to another, but whether the algorithm is *polynomial-time* or not does not depend on our choice of the computer. Thus, whatever reasonable class of computers we consider, a problem can be solved in polynomial-time on computers of this class if and only if we can solve it in polynomial time on a Turing machine (for details, see, e.g., Emde Boas [8]). Thus, if all we are interested in is whether an algorithm is feasible or not, Turing machines are quite sufficient.

Since we are also interested in more realistic estimates of computation time, we will use more realistic computer models.

RAM and bit complexity. In Turing machines, if we are currently facing cell 1, and we want to use the contents of cell n, we actually have to move step-by-step, and spend n computation steps just to reach this new cell. This is definitely not realistic. In real computers, if we know the number of the cell, we immediately go there. In other words, we can go to an arbitrarily ("randomly") chosen cell in a single step. If we add this ability to the Turing machine, we get the so-called *RAM* (Random Access Memory) computers. For the particular case when cells can only contain 0 and 1, the number of computational steps on RAM is sometimes called *bit complexity*.

Algebraic complexity. RAM is slightly more realistic than a Turing machine, but it is still not very realistic. In real-life computers, in addition to operations with bits, we have a hardware support for elementary *arithmetic* operations such as addition and multiplication of two integers. It is therefore reasonable, given an input of length n, to assume that we can perform addition and multiplication of integers of length $C \cdot n$ (for some reasonable C) in a single step. The number of computational steps on such machine is called *algebraic complexity*.

Algebraic complexity is the closest to the actual computation time and therefore, we will use this complexity measure in the paper. To be more precise:

- When we claim that something is *polynomial-time*, this claim (as we have already mentioned) will be independent on the choice of complexity measure. But:
- If we claim something more specific, like *linear time* (i.e., $t_U(n) \le C \cdot n$) or *quadratic time* (i.e., $t_U(n) \le C \cdot n^2$), we will mean exactly linear time (correspondingly, quadratic time) in the sense of algebraic complexity.

Comment. The relation between algebraic and bit complexity is analyzed, e.g., in Pan [55].

A remark about BSS complexity. To go from bit complexity to algebraic complexity, we counted each arithmetic operation with numbers of *fixed* length as a single computation step. We

can go further and count each operation with binary-rational numbers of *arbitrary* length (or even with arbitrary *real* numbers, not necessarily rational) as a single computation step. This definition was, in effect, proposed by Blum, Shub, and Smale [3] and is called BSS complexity (see also Smale [62], Meer [44]).

For our problems, BSS complexity is very useful in proving *negative* results: If a problem is *difficult* according to this BSS complexity, then it will be even more difficult if we only allow a narrower class of operations, i.e., it will be difficult according to algebraic complexity as well.

On the other hand, if a problem is *easy* in the sense of BSS complexity, it often means that this problem is actually easy, but sometimes, it can become difficult if we only allow operations with bounded numbers; examples of such *difference* between BSS and algebraic complexity are given, e.g., in Meer [43, 45].

Theoretically, the main problem of interval computations is algorithmically solvable. For rational functions, the problem of computing the range is, in principle, *algorithmically solvable*: namely, we can apply the so-called Tarski's algorithm [63]. However, this algorithm takes too long [6]: it sometimes takes time $\approx 2^{2^n}$ for an input of size n. As a result, even for small n, it may take billions of years. This is not a practical solution.

Main results [33]**.** If the function $f(x_1, \ldots, x_n)$ is itself difficult to compute, then, of course, it is difficult to compute the endpoints of the interval $\mathbf{y}$ even for *degenerate* input intervals $\mathbf{x}_i = [x_i, x_i]$.

To avoid this trivial situation, it makes sense to restrict ourselves to the simplest possible functions: functions that can be obtained by finitely many applications of the basic arithmetic operations (addition $+$, subtraction $-$, multiplication $\cdot$, and division $/$), i.e., to *rational* functions with rational coefficients.

The first result on complexity of interval computations was obtained by A. Gaganov in 1981: Gaganov proved that the main problem is computationally intractable (NP-hard) even for polynomials $f(x_1, \ldots, x_n)$.

Gaganov's result means that if we allow polynomials with arbitrarily many variables, of arbitrary degree, with arbitrary rational numbers for coefficients, and arbitrary intervals $\mathbf{x}_i$, then the problem is computationally intractable. A natural question is: what if we restrict some or all of these "arbitrary" parameters? E.g., what will happen if we only consider polynomials with bounded number of variables? of bounded degree? with bounded coefficients? with bounded (or even fixed) data intervals? It turns out that with most these restrictions, the problem is still NP-hard; the only exception is when we fix the number of variables; then, we get a polynomial-time algorithm.

Of all analyzed classes of polynomials, only for *linear* functions the basic problem has a simple and feasible algorithm. Since we cannot extend this result to more general polynomials, it is natural to extend it for different functions. For *fractionally* linear functions, the problem is still feasible; for a more general class of functions described by solutions of systems of linear equations, the problem is again NP-hard.

At first glance, it seems that all the above NP-hardness results paint a gloomy picture of computational intractability of data processing. The reality is, hopefully, not so gloomy: in spite of the *worst-case* complexity, some good heuristic algorithms are feasible in *almost all* cases (in some reasonable sense).

3 First Step Beyond Intervals: Error Estimation for Traditional Statistical Data Processing Algorithms under Interval Uncertainty

When we have n results $x_1, \ldots, x_n$ of repeated measurement of the same quantity (at different points, or at different moments of time), traditional statistical approach usually starts with computing their sample average $E = (x_1 + \ldots + x_n)/n$ and their (sample) variance

$$V = \frac{(x_1 - E)^2 + \ldots + (x_n - E)^2}{n}$$

(or, equivalently, the sample standard deviation $\sigma = \sqrt{V}$); see, e.g., [59].

In this section, we consider situations when we do not know the exact values of the quantities $x_1, \ldots, x_n$, we only know the intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$ of possible values of x_i. In such situations, for different possible values $x_i \in \mathbf{x}_i$, we get different values of E and V. The question is: what are the intervals $\mathbf{E}$ and $\mathbf{V}$ of possible values of E and V?

The practical importance of this question was emphasized, e.g., in [51, 52] on the example of processing geophysical data.

For E, the straightforward interval computations leads to the exact range:

$$\mathbf{E} = \frac{\mathbf{x}_1 + \ldots + \mathbf{x}_n}{n}, \text{ i.e., } \underline{E} = \frac{\underline{x}_1 + \ldots + \underline{x}_n}{n}, \text{ and } \overline{E} = \frac{\overline{x}_1 + \ldots + \overline{x}_n}{n}.$$

For V, straightforward interval computations lead to an excess width. For example, for $\mathbf{x}_1 = \mathbf{x}_2 = [0, 1]$, the variance is $V = (x_1 - x_2)^2/4$ and hence, the actual range $\mathbf{V} = [0, 0.25]$. On the other hand, $\mathbf{E} = [0, 1]$, hence

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

More sophisticated methods of interval computations also sometimes lead to an excess width.

Reason: in the formula for the average E, each variable only occurs once, and it is known that for such formulas, straightforward interval computations lead to the exact range (see, e.g., [19]). In the expression for variance, each variable x_i occurs several times: explicitly, in $(x_i - E)^2$, and explicitly, in the expression for E. In such cases, often, dependence between intermediate computation results leads to excess width of the results of straightforward interval computations. Not surprisingly, we do get excess width when applying straightforward interval computations to the above formula.

For variance, we can actually prove that the corresponding optimization problem is difficult:

Theorem 1. *Computing $\overline{V}$ is NP-hard.*

Proof. By definition, a problem is NP-hard if any problem from the class NP can be reduced to it. Therefore, to prove that a problem $\mathcal{P}$ is NP-hard, it is sufficient to reduce one of the known NP-hard problems $\mathcal{P}_0$ to $\mathcal{P}$.

In this case, since $\mathcal{P}_0$ is known to be NP-hard, this means that every problem from the class NP can be reduced to $\mathcal{P}_0$, and since $\mathcal{P}_0$ can be reduced to $\mathcal{P}$, thus, the original problem from the class NP is reducible to $\mathcal{P}$.

For our proof, as the known NP-hard problem $\mathcal{P}_0$, we take a *subset* problem: given n positive integers $s_1, \ldots, s_n$, to check whether there exist signs $\eta_i \in \{-1, +1\}$ for which the signed sum $\sum_{i=1}^{n} \eta_i \cdot s_i$ equals 0.

We will show that this problem can be reduced to the problem of computing $\overline{V}$, i.e., that to every instance $(s_1, \ldots, s_n)$ of the problem $\mathcal{P}_0$, we can put into correspondence such an instance of the $\overline{V}$-computing problem that based on its solution, we can easily check whether the desired signs exist.

As this instance, we take the instance corresponding to the intervals $[\underline{x}_i, \overline{x}_i] = [-s_i, s_i]$. We want to show that for the corresponding problem, $\overline{V} = C_0$, where we denoted $C_0 \stackrel{\text{def}}{=} \frac{1}{n} \cdot \sum\limits_{i=1}^{n} s_i^2$, if and only if there exist signs η_i for which $\sum \eta_i \cdot s_i = 0$.

1°. Let us first show that in all cases, $\overline{V} \le C_0$.

Indeed, it is known that the formula for the finite population variance can be reformulated in the following equivalent form:

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

Since $x_i \in [-s_i, s_i]$, we can conclude that $x_i^2 \le s_i^2$ hence $\sum x_i^2 \le \sum s_i^2$. Since $E^2 \ge 0$, we thus conclude that $V \le \frac{1}{n} \cdot \sum\limits_{i=1}^{n} s_i^2 = C_0$. In other words, every possible value V of the sample variance is smaller than or equal to C_0. Thus, the largest of these possible values, i.e., $\overline{V}$, also cannot exceed C_0, i.e., $\overline{V} \le C_0$.

2°. Let us now prove that if the desired signs η_i exist, then $\overline{V} = C_0$.

Indeed, in this case, for $x_i = \eta_i \cdot s_i$, we have $E = 0$ and $x_i^2 = s_i^2$, hence

$$V = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^2 = \frac{1}{n} \cdot \sum_{i=1}^{n} s_i^2 = C_0.$$

So, the variance V is always $\le C_0$, and it attains the value C_0 for some x_i. Therefore, $\overline{V} = C_0$.

3°. To complete the proof of Theorem 1, we must show that, vice versa, if $\overline{V} = C_0$, then the desired signs exist.

Indeed, let $\overline{V} = C_0$. The variance is a continuous function on a compact set $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$, hence its maximum on this compact set is attained for some values $x_1 \in \mathbf{x}_1 = [-s_1, s_1], \ldots, x_n \in \mathbf{x}_n = [-s_n, s_n]$. In other words, for the corresponding values of x_i, the variance V is equal to C_0.

Since $x_i \in [-s_i, s_i]$, we can conclude that $x_i^2 \le s_i^2$; since $E^2 \ge 0$, we get $V \le C_0$. If $|x_i|^2 < s_i^2$ or $E^2 > 0$, then we would have $\sigma^2 < C_0$. Thus, the only way to have $V = C_0$ is to have $x_i^2 = s_i^2$ and $E = 0$. The first equality leads to $x_i = \pm s_i$, i.e., to $x_i = \eta_i \cdot s_i$ for some $\eta_i \in \{-1, +1\}$. Since E is, by definition, the (arithmetic) average of the values x_i, the equality $E = 0$ then leads to $\sum\limits_{i=1}^{n} \eta_i \cdot s_i = 0$. So, if $\overline{V} = C_0$, then the desired signs do exist. The theorem is proven.

The very fact that computing the range of a quadratic function is NP-hard was first proven by Vavasis [66] (see also [33]). We have shown that this difficulty happens even for very simple quadratic functions frequently used in data processing.

A natural question is: maybe the difficulty comes from the requirement that the range be computed exactly? In practice, it is often sufficient to compute, in a reasonable amount of time, a usefully accurate estimate $\widetilde{\overline{V}}$ for $\overline{V}$, i.e., an estimate $\widetilde{\overline{V}}$ which is accurate with a given accuracy

$\varepsilon > 0$: $\left|\widetilde{\overline{V}} - \overline{V}\right| \le \varepsilon$. Alas, a simple modification of the above proof shows that for any ε, such computations are also NP-hard:

Theorem 2. *For every $\varepsilon > 0$, the problem of computing $\overline{V}$ with accuracy ε is NP-hard.*

It is worth mentioning that $\overline{V}$ can be computed exactly in exponential time $O(2^n)$:

Theorem 3. *There exists an algorithm that computes $\overline{V}$ in exponential time.*

Proof. Let $x_1^{(0)} \in \mathbf{x}_1, \ldots, x_n^{(0)} \in \mathbf{x}_n$ be the values for which the variance V attains maximum on the box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$.

Let us pick one of the n variables x_i, and let us fix the values of all the other variables x_j ($j \ne i$) at $x_j = x_j^{(0)}$. When we substitute $x_j = x_j^{(0)}$ for all $j \ne i$ into the expression for finite population variance, V becomes a quadratic function of x_i. This function of one variable should attain its maximum on the interval $\mathbf{x}_i$ at the value $x_i^{(0)}$.

By definition, the variance V is a sum of non-negative terms; thus, its value is always non-negative. Therefore, the corresponding quadratic function of one variable always has a global minimum. This function is decreasing before this global minimum, and increasing after it. Thus, its maximum on the interval $\mathbf{x}_i$ is attained at one of the endpoints of this interval.

In other words, for each variable x_i, the maximum is attained either for $x_i = \underline{x}_i$, or for $x_i = \overline{x}_i$. Thus, to find $\overline{V}$, it is sufficient to compute V for 2^n possible combinations $(x_1^{\pm}, \ldots, x_n^{\pm})$, where $x_i^- \stackrel{\text{def}}{=} \underline{x}_i$ and $x_i^+ \stackrel{\text{def}}{=} \overline{x}_i$, and find the largest of the resulting 2^n numbers. The theorem is proven.

For computing $\underline{V}$, there a feasible algorithm: specifically, our algorithm is *quadratic-time*, i.e., it requires $O(n^2)$ computational steps (arithmetic operations or comparisons) for n interval data points $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$.

The algorithm $\underline{\mathcal{A}}$ is as follows:

- First, we sort all $2n$ values $\underline{x}_i$, $\overline{x}_i$ into a sequence $x_{(1)} \le x_{(2)} \le \ldots \le x_{(2n)}$.
- Second, we compute $\underline{E}$ and $\overline{E}$ and select all "small intervals" $[x_{(k)}, x_{(k+1)}]$ that intersect with $[\underline{E}, \overline{E}]$.
- For each of the selected small intervals $[x_{(k)}, x_{(k+1)}]$, we compute the ratio $r_k = S_k/N_k$, where

$$S_k \stackrel{\text{def}}{=} \sum_{i:\underline{x}_i \ge x_{(k+1)}} \underline{x}_i + \sum_{j:\overline{x}_j \le x_{(k)}} \overline{x}_j,$$

  and N_k is the total number of such i's and j's If $r_k \in [x_{(k)}, x_{(k+1)}]$, then we compute

$$V_k \stackrel{\text{def}}{=} \frac{1}{n} \cdot \left(\sum_{i:\underline{x}_i \ge x_{(k+1)}} (\underline{x}_i - r_k)^2 + \sum_{j:\overline{x}_j \le x_{(k)}} (\overline{x}_j - r_k)^2 \right).$$

  If $N_k = 0$, we take $V_k \stackrel{\text{def}}{=} 0$.
- Finally, we return the smallest of the values V_k as $\underline{V}$.

Theorem 4. *The algorithm $\underline{\mathcal{A}}$ always compute $\underline{V}$ is quadratic time.*

Proof. Let us first show that this algorithm is indeed correct.

1°. Indeed, let $x_1^{(0)} \in \mathbf{x}_1, \ldots, x_n^{(0)} \in \mathbf{x}_n$ be the values for which the variance V attains minimum on the box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$.

Let us pick one of the n variables x_i, and let us fix the values of all the other variables x_j ($j \neq i$) at $x_j = x_j^{(0)}$. When we substitute $x_j = x_j^{(0)}$ for all $j \neq i$ into the expression for finite population variance, V becomes a quadratic function of x_i. This function of one variable should attain its minimum on the interval $\mathbf{x}_i$ at the value $x_i^{(0)}$.

As we have shown in the proof of Theorem 3, this function is decreasing before this global minimum, and increasing after it. This global minimum is attained when $\partial V/\partial x_i = 0$. Differentiating the formula that defines V with respect to x_i, we conclude that

$$\frac{\partial V}{\partial x_i} = \frac{1}{n} \cdot \left(2(x_i - E) + \sum_{j=1}^{n} 2(E - x_j) \cdot \frac{\partial E}{\partial x_j} \right).$$

Since $\partial E/\partial x_i = 1/n$, we conclude that

$$\frac{\partial V}{\partial x_i} = \frac{2}{n} \cdot \left((x_i - E) + \sum_{j=1}^{n} (E - x_j) \cdot \frac{1}{n} \right).$$

Here, $\sum(E - x_j) = n \cdot E - \sum x_j$. By definition of the average E, this difference is 0, hence the above formula takes the form $\partial V/\partial x_i = (2/n) \cdot (x_i - E)$. So, this function attains the minimum when $x_i - E = 0$, i.e., when $x_i = E$.

Since $E = (1/n) \cdot (x_i + \sum_i' x_j)$, where $\sum_i'$ means the sum over all $j \neq i$, the equality $x_i = E$ means that $x_i = x_i/n + (1/n) \cdot \sum_i' x_j^{(0)}$. Moving terms containing x_i into the left-hand side and dividing by the coefficient at x_i, we conclude that the minimum is attained when $x_i = E_i \stackrel{\text{def}}{=} \dfrac{1}{n-1} \cdot \sum_i' x_j^{(0)}$, i.e., when x_i is equal to the arithmetic average E_i of all other elements.

2°. Let us now use the knowledge of a *global* minimum to describe where the desired function attains its minimum on the interval $\mathbf{x}_i$.

In our general description of non-negative quadratic functions of one variable, we mentioned that each such function is decreasing before the global minimum and increasing after it. Thus, for $x_i < E_i$, the function V is decreasing; for $x_i > E_i$, this function in increasing. Therefore:

- If $E_i \in \mathbf{x}_i$, the global minimum of the function V of one variable is attained within the interval $\mathbf{x}_i$, hence the minimum on the interval $\mathbf{x}_i$ is attained for $x_i = E_i$.
- If $E_i < \underline{x}_i$, the function V is increasing on the interval $\mathbf{x}_i$ and therefore, its minimum on this interval is attained when $x_i = \underline{x}_i$.
- Finally, if $E_i > \overline{x}_i$, the function V is decreasing on the interval $\mathbf{x}_i$ and therefore, its minimum on this interval is attained when $x_i = \overline{x}_i$.

3°. Let us reformulate the above conditions in terms of the average

$$E = \frac{1}{n} \cdot x_i + \frac{n-1}{n} \cdot E_i.$$

- In the first case, when $x_i = E_i$, we have $x_i = E = E_i$, so $E \in \mathbf{x}_i$.
- In the second case, we have $E_i < \underline{x}_i$ and $x_i = \underline{x}_i$. Therefore, in this case, $E < \underline{x}_i$.

- In the third case, we have $E_i > \overline{x}_i$ and $x_i = \overline{x}_i$. Therefore, in this case, $E > \underline{x}_i$.

Thus:

- If $E \in \mathbf{x}_i$, then we cannot be in the second or third cases. Thus, we are in the first case, hence $x_i = E$.
- If $E < \underline{x}_i$, then we cannot be in the first or the third cases. Thus, we are the second case, hence $x_i = \underline{x}_i$.
- If $E > \overline{x}_i$, then we cannot be in the first or the second cases. Thus, we are in the third case, hence $x_i = \overline{x}_i$.

4°. So, as soon as we determine the position of E with respect to all the bounds $\underline{x}_i$ and $\overline{x}_i$, we will have a pretty good understanding of all the values x_i at which the minimum is attained. Hence, to find the minimum, we will analyze how the endpoints $\underline{x}_i$ and $\overline{x}_i$ divide the real line, and consider all the resulting sub-intervals.

Let the corresponding subinterval $[x_{(k)}, x_{(k+1)}]$ by fixed. For the i's for which $E \notin \mathbf{x}_i$, the values x_i that correspond to the minimal finite population variance are uniquely determined by the above formulas.

For the i's for which $E \in \mathbf{x}_i$ the selected value x_i should be equal to E. To determine this E, we can use the fact that E is equal to the average of all thus selected values x_i, in other words, that we should have

$$E = \frac{1}{n} \cdot \left(\sum_{i:\underline{x}_i \geq x_{(k+1)}} \underline{x}_i + (n - N_k) \cdot E + \sum_{j:\overline{x}_j \leq x_{(k)}} \overline{x}_j \right),$$

where $(n - N_k) \cdot E$ combines all the points for which $E \in \mathbf{x}_i$. Multiplying both sides of this equality by n and subtracting $n \cdot E$ from both sides, we conclude that $E = S_k/N_k$ – what we denoted, in the algorithm's description, by r_k. If thus defined r_k does not belong to the subinterval $[x_{(k)}, x_{(k+1)}]$, this contradiction with our initial assumption shows that there cannot be any minimum in this subinterval, so this subinterval can be easily dismissed.

The corresponding variance is denoted by V_k. If $N_k = 0$, this means that E belongs to all the intervals $\mathbf{x}_i$ and therefore, that the lower endpoint $\underline{V}$ is exactly 0 – so we assign $V_k = 0$. So, the algorithm is indeed correct.

5°. To complete the proof of the theorem, we must show that this algorithm indeed requires quadratic time. Indeed, sorting requires $O(n \cdot \log(n))$ steps (see, e.g., [5]), and the rest of the algorithm requires linear time ($O(n)$) for each of $2n$ subintervals, i.e., the total quadratic time. The theorem is proven.

NP-hardness of computing $\overline{V}$ means, crudely speaking, that there are no general ways for solving all particular cases of this problem (i.e., computing $\overline{V}$) in reasonable time.

However, we show that there are algorithms for computing $\overline{V}$ for many reasonable situations. Namely, we propose an efficient algorithm that computes $\overline{V}$ for the case when all the interval midpoints ("measured values") $\widetilde{x}_i = (\underline{x}_i + \overline{x}_i)/2$ are definitely different from each other, in the sense that the "narrowed" intervals $[\widetilde{x}_i - \Delta_i/n, \widetilde{x}_i + \Delta_i/n]$ – where $\Delta_i = (\underline{x}_i - \overline{x}_i)/2$ is the interval's half-width – do not intersect with each other.

This algorithm $\overline{\mathcal{A}}$ is as follows:

- First, we sort all $2n$ endpoints of the narrowed intervals $\widetilde{x}_i - \Delta_i/n$ and $\widetilde{x}_i + \Delta_i/n$ into a sequence $x_{(1)} \le x_{(2)} \le \ldots \le x_{(2n)}$. This enables us to divide the real line into $2n+1$ segments ("small intervals") $[x_{(k)}, x_{(k+1)}]$, where we denoted $x_{(0)} \stackrel{\text{def}}{=} -\infty$ and $x_{(2n+1)} \stackrel{\text{def}}{=} +\infty$.
- Second, we compute $\underline{E}$ and $\overline{E}$ and pick all "small intervals" $[x_{(k)}, x_{(k+1)}]$ that intersect with $[\underline{E}, \overline{E}]$.
- For each of remaining small intervals $[x_{(k)}, x_{(k+1)}]$, for each i from 1 to n, we pick the following value of x_i:
  - if $x_{(k+1)} < \widetilde{x}_i - \Delta_i/n$, then we pick $x_i = \overline{x}_i$;
  - if $x_{(k)} > \widetilde{x}_i + \Delta_i/n$, then we pick $x_i = \underline{x}_i$;
  - for all other i, we consider both possible values $x_i = \overline{x}_i$ and $x_i = \underline{x}_i$.

  As a result, we get one or several sequences of x_i. For each of these sequences, we check whether the average E of the selected values $x_1, \ldots, x_n$ is indeed within this small interval, and if it is, compute the variance by using the formula that defines V.
- Finally, we return the largest of the computed variances as $\overline{V}$.

Theorem 5. *The algorithm $\overline{\mathcal{A}}$ computes $\overline{V}$ is quadratic time for all the cases in which the "narrowed" intervals do not intersect with each other.*

This algorithm also works when, for some fixed k, no more than k "narrowed" intervals can have a common point:

Theorem 6. *For every positive integer k, the algorithm $\overline{\mathcal{A}}$ computes $\overline{V}$ is quadratic time for all the cases in which no more than k "narrowed" intervals can have a common point.*

Proof. Let us first show that this algorithm is indeed correct.

1°. Similarly to the proof of Theorem 4, let $x_1, \ldots, x_n$ be the values at which the finite population variance attain its maximum on the box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$. If we fix the values of all the variables but one x_i, then V becomes a quadratic function of x_i. When the function V attains maximum over $x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n$, then this quadratic function of one variable will attain its maximum on the interval $\mathbf{x}_i$ at the point x_i.

We have already shown, in the proof of Theorem 4, that this quadratic function has a (global) minimum at $x_i = E_i$. Since this quadratic function of one variable is always non-negative, it cannot have a global maximum. Therefore, its maximum on the interval $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ is attained at one of the endpoints of this interval.

An arbitrary quadratic function of one variable is symmetric with respect to the location of its global minimum, so its maximum on any interval is attained at the point which is the farthest from the minimum. There is exactly one point which is equally close to both endpoints of the interval $\mathbf{x}_i$: its midpoint $\widetilde{x}_i$. Depending on whether the global minimum is to the left, to the right, or exactly at the midpoint, we get the following three possible cases:

1. If the global minimum E_i is to the left of the midpoint $\widetilde{x}_i$, i.e., if $E_i < \widetilde{x}_i$, then the upper endpoint is the farthest from E_i. In this case, the maximum of the quadratic function is attained at its upper endpoint, i.e., $x_i = \overline{x}_i$.

2. Similarly, if the global minimum E_i is to the right of the midpoint $\widetilde{x}_i$, i.e., if $E_i > \widetilde{x}_i$, then the lower endpoint is the farthest from E_i. In this case, the maximum of the quadratic function is attained at its lower endpoint, i.e., $x_i = \underline{x}_i$.

3. If $E_i = \widetilde{x}_i$, then the maximum of V is attained at both endpoints of the interval $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$.

2°. In the third case, we have either $x_i = \underline{x}_i$ or $x_i = \overline{x}_i$. Depending on whether x_i is equal to the lower or to the upper endpoints, we can "combine" the corresponding situations with Cases 1 and 2. As a result, we arrive at the conclusion that one of the following two situations happen:

1. either $E_i \le \widetilde{x}_i$ and $x_i = \overline{x}_i$;

2. either $E_i \ge \widetilde{x}_i$ and $x_i = \underline{x}_i$.

3°. Similarly to the proof of Theorem 4, let us reformulate these conclusions in terms of the average E of the maximizing values $x_1, \ldots, x_n$.

By definition, $E_i = \dfrac{1}{n-1} \cdot \sum_i' x_j$, and $\sum_j' x_j = \sum_j x_j - x_i$. By definition of E, we have $\displaystyle\sum_j x_j = n \cdot E$, therefore, $E_i = \dfrac{n \cdot E - x_i}{n-1}$. Let us apply this formula to the above three cases.

In the first case, we have $\widetilde{x}_i \ge E_i$. So, in terms of E, we get the inequality $\widetilde{x}_i \ge \dfrac{n \cdot E - x_i}{n-1}$. Multiplying both sides of this inequality by $n-1$, and using the fact that in this case, $x_i = \overline{x}_i = \widetilde{x}_i + \Delta_i$, we conclude that $(n-1) \cdot \widetilde{x}_i \ge n \cdot E - \widetilde{x}_i - \Delta_i$. Moving all the terms but $n \cdot E$ to the left-hand side and dividing by E, we get the following inequality: $E \le \widetilde{x}_i + \Delta_i/n$.

In the second case, we similarly get the inequality $E \le \widetilde{\Delta}_i/n$. So:

- In Case 1, we have $E \le \widetilde{x}_i + \Delta_i/n$ and $x_i = \overline{x}_i$.

- In Case 2, we have $E \ge \widetilde{x}_i - \Delta_i/n$ and $x_i = \underline{x}_i$.

Therefore:

- If $E < \widetilde{x}_i - \Delta_i/n$, this means that we cannot be in Case 2. So we must be in Case 1 and therefore, we must have $x_i = \overline{x}_i$.

- If $E > \widetilde{x}_i + \Delta_i/n$, this means that we cannot be in Case 1. So, we must be in Case 2 and therefore, we must have $x_i = \underline{x}_i$.

The only case when we do not know which endpoint for x_i we should choose is the case when E belongs to the narrowed interval $[\widetilde{x}_i - \Delta/n, \widetilde{x}_i + \Delta_i]$.

4°. Hence, once we know where E is with respect to the endpoints of all narrowed intervals, we can determine the values of all optimal x_i – except for those that are within this narrowed interval. Since we consider the case when no more than k narrowed intervals can have a common point, we have no more than k undecided values x_i. Trying all possible combinations of lower and upper endpoints for these $\le k$ values requires $\le 2^k$ steps.

Thus, the overall number of steps is $O(2^k \cdot n^2)$. Since k is a constant, the overall number of steps is thus $O(n^2)$. The theorem is proven.

4 Important Example: Interval Computations Related to Privacy in Statistical Databases

Need for privacy. Privacy is an important issue in the statistical analysis of human-related data. For example, to check whether in a certain geographic area, there is a gender-based discrimination, we can use the census data to check, e.g., whether for all people from this area who have the same same level of education, there is a correlation between salary and gender. One can think of numerous possible questions of this type related to different sociological, political, medical, economic, and other questions. From this viewpoint, it is desirable to give researches *ability to perform* whatever *statistical analysis* of this data that is reasonable for their specific research.

On the other hand, we do not want to give them direct access to the raw census data, because a large part of the census data is *confidential*. For example, for most people (those who work in private sector) salary information is confidential. Suppose that a corporation is deciding where to built a new plant and has not yet decided between two possible areas. This corporation would benefit from knowing the average salary of people of needed education level in these two areas, because this information would help them estimate how much it will cost to bring local people on board. However, since salary information is confidential, the company should not be able to know the exact salaries of different potential workers.

The need for privacy is also extremely important for *medical* experiments, where we should be able to make statistical conclusions about, e.g., the efficiency of a new medicine without disclosing any potentially embarrassing details from the individual medical records.

Such databases in which the outside users cannot access individual records but can solicit statistical information are often called *statistical databases*.

How privacy is protected now and why it is not always sufficient. At present, one of the main (and most efficient) methods of protecting privacy in databases is the disaggregation of the data: instead of keeping a record with all the information about a person, we divide this record into several subrecords. For example, instead of keeping a single census record about a female professor leaving in New Jersey with three cats, we split this record into several subrecords: a subrecord about a person living in New Jersey with three cats (this subrecord will be useful for pet statistics), a subrecord about a female professor living in New Jersey (this subrecord will be useful for gender-based employment statistics), etc.

Such disaggregation helps to protect privacy. Indeed, if we keep the original full records, then we can narrow down a request in such a way that only one person will qualify: e.g., a request about the average salary of all female professors living in New Jersey with three cats etc. will eventually lead to an actual salary of that person. On the other hand, when records are disaggregated, whatever query we ask, be it an average salary of all New Jersey residents with exactly three cats or an average salary of all female professors from New Jersey, we will most likely not narrow down to a single person.

Disaggregation is very useful for protecting privacy, but it is not sufficient. Indeed, suppose that we keep a university salary database; for privacy protection, we keep all the records anonymous so the only information in the database is the actual salary values. What happens if we allow all possible statistical queries, including queries like "How many people have salary 83.6K or smaller"? By asking appropriate queries, we can find the salary values close to which the answer changes – and these values are exactly the actual salaries from the database. Thus, if we know that, e.g., the university president is the highest-paying professor, we will be able to get her salary as the largest of these actual salaries. How can we avoid this privacy violation?

Privacy leads to intervals. A natural way to fully describe a single real-valued random variable η is to provide the values of its cumulative density function (CDF)

$$F(x) = \text{Prob}(\eta \le x)$$

for all possible real numbers x. Once we know $F(x)$, we can determine the values of all possible statistical characteristics of this random variable – e.g., its first moment, second moment, variance, etc. Thus, it is natural to allow the users to solicit the values of $F(x)$ for different x; from this information, the users will be able to reconstruct all other statistical characteristics.

For discrete data $x_1, \ldots, x_n$, the corresponding sample distribution – in which each value x_i occurs with probability $1/n$ – is described by the CDF $F(x)$ for which

$$F(x) = (1/n) \cdot \#\{i : x_i \le x\}.$$

To get the full information about the data, we should allow the user to ask for the values $F(x)$ for all possible real numbers x. However, as we have mentioned, once we know the values $F(x)$ for all x, we can determine all the values x_i. Thus, if we want to keep privacy, we must only allow the users to know $F(x)$ for some fixed values $x^{(1)} \le \ldots \le x^{(m)}$. This way, instead of the actual values x_i, all we know is an *interval* $[x^{(k)}, x^{(k+1)}]$ that contains x_i. Intervals corresponding to different values are *almost disjoint*, i.e., either disjoint (intersect in at most one point) or identical. How can we compute statistical characteristics based on this information?

Theorem 7. *There exists a quadratic-time algorithm that computes the exact range* $\mathbf{V}$ *of the variance* V *for the case when intervals* $\mathbf{x}_i$ *of possible values of* x_i *are pairwise almost disjoint.*

Proof. Since there exists an algorithm that computes $\underline{V}$ in feasible time, it is sufficient to produce a feasible algorithm for computing $\overline{V}$.

According to the proof of Theorems 3, 5, and 6, the values $x_i \in \mathbf{x}_i$ that lead to the largest possible value of V satisfy the following property:

- if $E \le \underline{x}_i$, then $x_i = \overline{x}_i$;
- if $E \ge \overline{x}_i$, then $x_i = \underline{x}_i$;
- if $E \in (\underline{x}_i, \overline{x}_i)$, then $x_i = \underline{x}_i$ or $x_i = \overline{x}_i$.

In order to use this property to compute $\overline{V}$, we test all possible locations of E in relation to the intervals $\mathbf{x}_i$: $E = \underline{x}_i$, $E = \overline{x}_i$, and $E \in (\underline{x}_i, \overline{x}_i)$ for different $i = 1, 2, \ldots, n$.

Let us first consider the cases when $E = \underline{x}_i$ (the case when $E = \overline{x}_i$ is treated similarly). In these cases, since the intervals $\mathbf{x}_i$ are almost disjoint, the above property uniquely determines the values x_i; thus, we can compute E, check whether it indeed satisfies the corresponding condition, and if yes, compute the corresponding value V.

Let us now consider the cases when $E \in (\underline{x}_i, \overline{x}_i)$. Let k denote the number of different intervals of such type, and let n_j, $j = 1, \ldots, k$ denote the number of intervals $\mathbf{x}_i$ that coincide with j-th interval. Then, $n = n_1 + \ldots + n_k$. For each of these k intervals $\mathbf{x}_j$, the values of x_i are uniquely determined when $\overline{x}_j \le \underline{x}_i$ or $\overline{x}_i \le \underline{x}_j$; for the remaining n_j values x_i, we have $x_i = \underline{x}_i$ or $x_i = \overline{x}_i$. Modulo transposition, the resulting set of values $\{x_1, \ldots, x_n\}$ is uniquely determined by how many of these n_j x_i's are equal to $\overline{x}_i$. The number of such x_i's can be 0, 1, 2,..., $n_j + 1$. Thus, the total number of such combinations is equal to $n_j + 1$. Overall, for all j from 1 to k, we have $\sum_{j=1}^{k}(n_j + 1) = \sum_{j=1}^{k} n_j + k = n + k \le 2n$ resulting sets $\{x_1, \ldots, x_n\}$. For each of these sets, we

compute E, check that the resulting E is indeed inside the corresponding interval $\mathbf{x}_i$, and if it is, we compute V.

Thus, we have $\leq 2n + n = 3n$ cases, for each of which we need $O(n)$ computations to compute V. The largest of these V is the desired $\overline{V}$, and we compute it in time $\leq 3n \cdot O(n) = O(n^2)$. The proposition is proven.

Comment. Similar algorithms can be provided for computing the exact range of covariance between two interval-valued data sequences; in general, the problem of computing the range for covariance is NP-hard [54].

5 Second Step Beyond Intervals: Extension of Interval Arithmetic to Situations with Partial Information about Probabilities

Practical problem. In some practical situations, in addition to the lower and upper bounds on each random variable x_i, we know the bounds $\mathbf{E}_i = [\underline{E}_i, \overline{E}_i]$ on its mean E_i.

Indeed, in measurement practice (see, e.g., [59]), the overall measurement error Δx is usually represented as a sum of two components:

- a *systematic* error component $\Delta_s x$ which is defined as the expected value $E[\Delta x]$, and
- a *random* error component $\Delta_r x$ which is defined as the difference between the overall measurement error and the systematic error component: $\Delta_r x \stackrel{\text{def}}{=} \Delta x - \Delta_s x$.

In addition to the bound Δ on the overall measurement error, the manufacturers of the measuring instrument often provide an upper bound Δ_s on the systematic error component: $|\Delta_s x| \leq \Delta_s$.

This additional information is provided because, with this additional information, we not only get a bound on the accuracy of a single measurement, but we also get an idea of what accuracy we can attain if we use repeated measurements to increase the measurement accuracy. Indeed, the very idea that repeated measurements can improve the measurement accuracy is natural: we measure the same quantity by using the same measurement instrument several (N) times, and then take, e.g., an arithmetic average $\bar{x} = (\widetilde{x}^{(1)} + \ldots + \widetilde{x}^{(N)})/N$ of the corresponding measurement results $\widetilde{x}^{(1)} = x + \Delta x^{(1)}, \ldots, \widetilde{x}^{(N)} = x + \Delta x^{(N)}$.

- If systematic error is the only error component, then all the measurements lead to exactly the same value $\widetilde{x}^{(1)} = \ldots = \widetilde{x}^{(N)}$, and averaging does not change the value – hence does not improve the accuracy.
- On the other hand, if we know that the systematic error component is 0, i.e., $E[\Delta x] = 0$ and $E[\widetilde{x}] = x$, then, as $N \to \infty$, the arithmetic average tends to the actual value x. In this case, by repeating the measurements sufficiently many times, we can determine the actual value of x with an arbitrary given accuracy.

In general, by repeating measurements sufficiently many times, we can arbitrarily decrease the random error component and thus attain accuracy as close to Δ_s as we want.

When this additional information is given, then, after we performed a measurement and got a measurement result $\widetilde{x}$, then not only we get the information that the actual value x of the measured quantity belongs to the interval $\mathbf{x} = [\widetilde{x} - \Delta, \widetilde{x} + \Delta]$, but we can also conclude that the expected value of $x = \widetilde{x} - \Delta x$ (which is equal to $E[x] = \widetilde{x} - E[\Delta x] = \widetilde{x} - \Delta_s x$) belongs to the interval $\mathbf{E} = [\widetilde{x} - \Delta_s, \widetilde{x} + \Delta_s]$.

If we have this information for every x_i, then, in addition to the interval $\mathbf{y}$ of possible value of y, we would also like to know the interval of possible values of $E[y]$. This additional interval will hopefully provide us with the information on how repeated measurements can improve the accuracy of this indirect measurement. Thus, we arrive at the following problem.

Resulting optimization problem. In more optimization terms, we want to solve the following problem: given an algorithm computing a function $f(x_1, \ldots, x_n)$ from R^n to R; and values $\underline{x}_1$, $\overline{x}_1$, $\ldots$, $\underline{x}_n$, $\overline{x}_n$, $\underline{E}_1$, $\overline{E}_1$, $\ldots$, $\underline{E}_n$, $\overline{E}_n$, we want to find

$$\underline{E} \stackrel{\text{def}}{=} \min\{E[f(x_1, \ldots, x_n)] \mid \text{ all distributions of } (x_1, \ldots, x_n) \text{ for which}$$

$$x_1 \in [\underline{x}_1, \overline{x}_1], \ldots, x_n \in [\underline{x}_n, \overline{x}_n], E[x_1] \in [\underline{E}_1, \overline{E}_1], \ldots E[x_n] \in [\underline{E}_n, \overline{E}_n]\};$$

and $\overline{E}$ which is the maximum of $E[f(x_1, \ldots, x_n)]$ for all such distributions.

In addition to considering all possible distributions, we can also consider the case when all the variables x_i are independent.

Analog of straightforward interval computations. The main idea behind straightforward interval computations can be applied here as well. Namely, first, we find out how to solve this problem for the case when $n = 2$ and $f(x_1, x_2)$ is one of the standard arithmetic operations. Then, once we have an arbitrary algorithm $f(x_1, \ldots, x_n)$, we parse it and replace each elementary operation on real numbers with the corresponding operation on quadruples $(\underline{x}, \underline{E}, \overline{E}, \overline{x})$.

To implement this idea, we must therefore know how to, solve the above problem for elementary operations.

Solution. For *addition*, the answer is simple. Since $E[x_1 + x_2] = E[x_1] + E[x_2]$, if $y = x_1 + x_2$, there is only one possible value for $E = E[y]$: the value $E = E_1 + E_2$. This value does not depend on whether we have correlation or nor, and whether we have any information about the correlation. Thus, $\mathbf{E} = \mathbf{E}_1 + \mathbf{E}_2$.

Similarly, the answer is simple for *subtraction:* if $y = x_1 - x_2$, there is only one possible value for $E = E[y]$: the value $E = E_1 - E_2$. Thus, $\mathbf{E} = \mathbf{E}_1 - \mathbf{E}_2$.

For *multiplication*, if the variables x_1 and x_2 are independent, then $E[x_1 \cdot x_2] = E[x_1] \cdot E[x_2]$. Hence, if $y = x_1 \cdot x_2$ and x_1 and x_2 are independent, there is only one possible value for $E = E[y]$: the value $E = E_1 \cdot E_2$; hence $\mathbf{E} = \mathbf{E}_1 \cdot \mathbf{E}_2$.

The first non-trivial case is the case of multiplication in the presence of possible correlation. When we know the exact values of E_1 and E_2, the solution to the above problem is as follows:

Theorem 8. *For multiplication $y = x_1 \cdot x_2$, when we have no information about the correlation,*

$$\underline{E} = \max(p_1 + p_2 - 1, 0) \cdot \overline{x}_1 \cdot \overline{x}_2 + \min(p_1, 1 - p_2) \cdot \overline{x}_1 \cdot \underline{x}_2 + \min(1 - p_1, p_2) \cdot \underline{x}_1 \cdot \overline{x}_2 +$$

$$\max(1 - p_1 - p_2, 0) \cdot \underline{x}_1 \cdot \underline{x}_2;$$

and

$$\overline{E} = \min(p_1, p_2) \cdot \overline{x}_1 \cdot \overline{x}_2 + \max(p_1 - p_2, 0) \cdot \overline{x}_1 \cdot \underline{x}_2 + \max(p_2 - p_1, 0) \cdot \underline{x}_1 \cdot \overline{x}_2 +$$

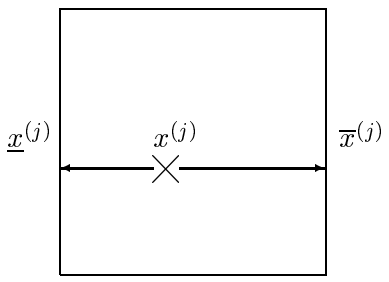
$$\min(1 - p_1, 1 - p_2) \cdot \underline{x}_1 \cdot \underline{x}_2,$$

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

Proof. Let us show that a general distribution with $E[x_i] = E_i$ can be simplified without changing the values $E[x_i]$ and $E[x_1 \cdot x_2]$. Thus, to describe possible values of $E[x_1 \cdot x_2]$, we do not need to consider all possible distributions, it is sufficient to consider only the simplified ones.

We will describe the simplification for discrete distributions that concentrate on finitely many points $x^{(j)} = (x_1^{(j)}, x_2^{(j)})$, $1 \le j \le N$. An arbitrary probability distribution can be approximated by such distributions, so we do not lose anything by this restriction.

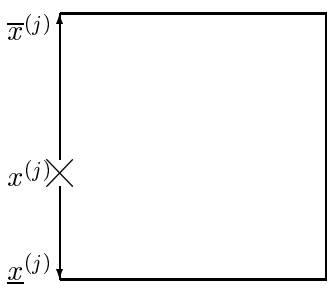
So, we have a probability distribution in which the point $x^{(1)}$ appears with the probability $p^{(1)}$, the point $x^{(2)}$ appears with the probability $p^{(2)}$, etc. Let us modify this distribution as follows: pick a point $x^{(j)} = (x_1^{(j)}, x_2^{(j)})$ that occurs with probability $p^{(j)}$, and replace it with two points: $\overline{x}^{(j)} = (\overline{x}_1, x_2^{(j)})$ with probability $p^{(j)} \cdot \overline{p}^{(j)}$ and $\underline{x}^{(j)} = (\underline{x}_1, x_2^{(j)})$ with probability $p^{(j)} \cdot \underline{p}^{(j)}$, where $\overline{p}^{(j)} \stackrel{\text{def}}{=} (x_1^{(j)} - \underline{x}_1)/(\overline{x}_1 - \underline{x}_1)$ and $\underline{p}^{(j)} \stackrel{\text{def}}{=} 1 - \overline{p}^{(j)}$:



Here, the values $\overline{p}^{(j)}$ and $\underline{p}^{(j)} = 1 - \overline{p}^{(j)}$ are chosen in such a way that $\overline{p}^{(j)} \cdot \overline{x}_1 + \underline{p}^{(j)} \cdot \underline{x}_1 = x_1^{(j)}$. Due to this choice, $p^{(j)} \cdot \overline{p}^{(j)} \cdot \overline{x}_1 + p^{(j)} \cdot \underline{p}^{(j)} \cdot \underline{x}_1 = p^{(j)} \cdot x_1^{(j)}$, hence for the new distribution, the mathematical expectation $E[x_1]$ is the same as for the old one. Similarly, we can prove that the values $E[x_2]$ and $E[x_1 \cdot x_2]$ do not change.

We started with a general discrete distribution with N points for each of which $x_1^{(j)}$ could be inside the interval $\mathbf{x}_1$, and we have a new distribution for which $\le N - 1$ points have the value x_1 inside this interval. We can perform a similar replacement for all N points and get a distribution with the same values of $E[x_1]$, $E[x_2]$, and $E[x_1 \cdot x_2]$ as the original one but for which, for every point, x_1 is equal either to $\underline{x}_1$, or to $\overline{x}_1$.

For the new distribution, we can perform a similar transformation relative to x_1 and end up – without changing the values x_1 – with the distribution for which always either $x_2 = \underline{x}_1$ or $x_2 = \overline{x}_2$:



Thus, instead of considering all possible distributions, it is sufficient to consider only distributions for which $x_1 \in \{\underline{x}_1, \overline{x}_1\}$ and $x_2 \in \{\underline{x}_2, \overline{x}_2\}$. In other words, it is sufficient to consider only

distributions which are located in the four corner points $(\underline{x}_1, \underline{x}_2)$, $(\underline{x}_1, \overline{x}_2)$, $(\overline{x}_1, \underline{x}_2)$, and $(\overline{x}_1, \overline{x}_2)$ of the box $\mathbf{x}_1 \times \mathbf{x}_2$.

Such distribution can be characterized by the probabilities of these four points. These four probabilities must satisfy 3 conditions: that their sum is 1, that $E[x_1]$ is E_1, and that $E[x_2] = E_2$. Thus, we only have one parameter left; optimizing with respect to this parameter, we get the desired formulas for $\underline{E}$ and $\overline{E}$. The theorem is proven.

When we only know the intervals $\mathbf{E}_i$ of possible values of E_i, instead of the values p_i, we have the corresponding intervals $\mathbf{p}_i = (\mathbf{E}_i - \underline{x}_i)/(\overline{E}_i - \underline{x}_i)$. In terms of these intervals, we get the following results:

Theorem 9. *For multiplication under no information about dependence, to find $\underline{E}$, it is sufficient to consider the following combinations of p_1 and p_2:*

- $p_1 = \underline{p}_1$ *and* $p_2 = \underline{p}_2$*;* $p_1 = \underline{p}_1$ *and* $p_2 = \overline{p}_2$*;* $p_1 = \overline{p}_1$ *and* $p_2 = \underline{p}_2$*;* $p_1 = \overline{p}_1$ *and* $p_2 = \overline{p}_2$*;*
- $p_1 = \max(\underline{p}_1, 1 - \overline{p}_2)$ *and* $p_2 = 1 - p_1$ *(if* $1 \in \mathbf{p}_1 + \mathbf{p}_2$*); and*
- $p_1 = \min(\overline{p}_1, 1 - \underline{p}_2)$ *and* $p_2 = 1 - p_1$ *(if* $1 \in \mathbf{p}_1 + \mathbf{p}_2$*).*

The smallest value of $\underline{E}$ for all these cases is the desired lower bound $\underline{E}$.

Theorem 10. *For multiplication under no information about dependence, to find $\overline{E}$, it is sufficient to consider the following combinations of p_1 and p_2:*

- $p_1 = \underline{p}_1$ *and* $p_2 = \underline{p}_2$*;* $p_1 = \underline{p}_1$ *and* $p_2 = \overline{p}_2$*;* $p_1 = \overline{p}_1$ *and* $p_2 = \underline{p}_2$*;* $p_1 = \overline{p}_1$ *and* $p_2 = \overline{p}_2$*;*
- $p_1 = p_2 = \max(\underline{p}_1, \underline{p}_2)$ *(if* $\mathbf{p}_1 \cap \mathbf{p}_2 \neq \emptyset$*); and*
- $p_1 = p_2 = \min(\overline{p}_1, \overline{p}_2)$ *(if* $\mathbf{p}_1 \cap \mathbf{p}_2 \neq \emptyset$*).*

The largest value of $\overline{E}$ for all these cases is the desired upper bound $\overline{E}$.

Proof. We will prove Theorem 10; the proof of Theorem 9 is similar. The formula for $\overline{E}$ given in Theorem 8 can be simplified if we consider two cases: $p_1 \leq p_2$ and $p_1 \geq p_2$. To find the largest possible value $\overline{E}$ of E, it is sufficient to consider the largest possible values for each of these cases, and then take the largest of the resulting two numbers.

In each case, for a fixed p_2, the formula is linear in p_1. To find the maximum of a linear function on an interval, it is sufficient to consider this interval's endpoints. Thus, the maximum in p_1 is attained when either p_1 attains its smallest possible value $\underline{p}_1$, or when p_1 attains the largest possible value within this case; depending on p_2, this value is either $p_1 = \overline{p}_1$ or $p_1 = p_2$.

Thus, to find the maximum for each cases, it is sufficient to consider only the following cases: $p_1 = \underline{p}_1$, $p_1 = \overline{p}_1$, and $p_1 = p_2$. Similarly, it is sufficient to consider only the following cases for p_2: $p_2 = \underline{p}_2$, $p_2 = \overline{p}_2$, and $p_1 = p_2$.

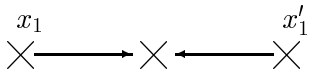
When $p_1 = p_2$, the probability $p_1 = p_2$ can take all possible values from the intersection $\mathbf{p}_1 \cap \mathbf{p}_2$. the formula for $\overline{E}$ is linear in p_1, so to find its maximum, it is sufficient to consider the endpoints of the interval $\mathbf{p}_1 \cap \mathbf{p}_2$, i.e., the values $p_1 = p_2 = \max(\underline{p}_1, \underline{p}_2)$ and $p_1 = p_2 = \min(\overline{p}_1, \overline{p}_2)$. The theorem is proven.

For the *inverse* $y = 1/x_1$, the finite range is possible only when $0 \notin \mathbf{x}_1$. Without losing generality, we can consider the case when $0 < \underline{x}_1$. In this case, methods presented in [60] lead to the following bound:

Theorem 11. *For the inverse $y = 1/x_1$, the range of possible values of E is* $\mathbf{E} = [1/E_1, p_1/\overline{x}_1 + (1-p_1)/\underline{x}_1]$.

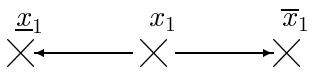
(Here p_1 denotes the same value as in Theorem 8).

Proof. For $x_1 > 0$, the function $f(x_1) \stackrel{\text{def}}{=} 1/x_1$ is convex: for every x_1, x_1', and $\alpha \in [0,1]$, we have $f(\alpha \cdot x_1 + (1-\alpha) \cdot x_1') \le \alpha \cdot f(x_1) + (1-\alpha) \cdot f(x_1')$. Hence, if we are looking for a minimum of $E[1/x_1]$, we can replace every two points from the probability distribution with their average, and the resulting value of $E[1/x_1]$ will only decrease:



So, the minimum is attained when the probability distribution is concentrated on a single value – which has to be E_1. Thus, the smallest possible value of $E[1/x_1]$ is $1/E_1$.

Due to the same convexity, if we want maximum of $E[1/x_1]$, we should replace every value $x_1 \in [\underline{x}_1, \overline{x}_1]$ by a probabilistic combination of the values $\underline{x}_1, \overline{x}_1$:



So, the maximum is attained when the probability distribution is concentrated on these two end-points $\underline{x}_1$ and $\overline{x}_1$. Since the average of x_1 should be equal to E_1, we can, similarly to the proof of Theorem 8, conclude that in this distribution, $\overline{x}_1$ occurs with probability p_1, and $\underline{x}_1$ occurs with probability $1 - p_1$. For this distribution, the value $E[1/x_1]$ is exactly the upper bound from the formulation of the theorem. The theorem is proven.

Theorem 12. *For minimum $y = \min(x_1, x_2)$, when x_1 and x_2 are independent, we have $\overline{E} = \min(E_1, E_2)$ and*

$$\underline{E} = p_1 \cdot p_2 \cdot \min(\overline{x}_1, \overline{x}_2) + p_1 \cdot (1-p_2) \cdot \min(\overline{x}_1, \underline{x}_2) + (1-p_1) \cdot p_2 \cdot \min(\underline{x}_1, \overline{x}_2) +$$
$$(1-p_1) \cdot (1-p_2) \cdot \min(\underline{x}_1, \underline{x}_2).$$

Theorem 13. *For maximum $y = \min(x_1, x_2)$, when x_1 and x_2 are independent, we have $\underline{E} = \max(E_1, E_2)$ and*

$$\overline{E} = p_1 \cdot p_2 \cdot \max(\overline{x}_1, \overline{x}_2) + p_1 \cdot (1-p_2) \cdot \max(\overline{x}_1, \underline{x}_2) + (1-p_1) \cdot p_2 \cdot \max(\underline{x}_1, \overline{x}_2) +$$
$$(1-p_1) \cdot (1-p_2) \cdot \max(\underline{x}_1, \underline{x}_2).$$

Proof. We will prove Theorem 12; the proof of Theorem 13 is similar. Since $\min(x_1, x_2) \le x_1$, we have $E[\min(x_1, x_2)] \le E[x_1] = E_1$. Similarly, $E[\min(x_1, x_2)] \le E_2$, hence, $E[\min(x_1, x_2)] \le \min(E_1, E_2)$. The value $\min(E_1, E_2)$ is possible when $x_1 = E_1$ with probability 1 and $x_2 = E_2$ with probability 1. Thus, $\min(E_1, E_2)$ is the exact upper bound for $E[\min(x_1, x_2)]$.

For each x_2, the function $x_1 \to \min(x_1, x_2)$ is concave; therefore, if we replace each point $x^{(j)} = (x_1^{(j)}, x_2^{(j)})$ by the corresponding probabilistic combination of the points $(\underline{x}_1, x_2^{(j)})$ and $(\overline{x}_1, x_2^{(j)})$ (as in the proof of Theorem 11), we preserve $E[x_1]$ and $E[x_2]$ and decrease the value $E[\min(x_1, x_2)]$. Thus, when we are looking for the smallest possible value of $E[\min(x_1, x_2)]$, it is sufficient to

consider only the distributions for which x_1 is located at one of the endpoints $\underline{x}_1$ or $\overline{x}_1$. Similarly to the proof of Theorem 8, the probability of $\overline{x}_1$ is equal to p_1.

Similarly, we can conclude that to find the largest possible value of $E[\min(x_1, x_2)]$, it is sufficient to consider only distributions in which x_2 can take only two values: $\underline{x}_2$ and $\overline{x}_2$. To get the desired value of E_2, we must have $\overline{x}_2$ with probability p_1 and $\underline{x}_2$ with probability $1 - p_2$.

Since we consider the case when x_1 and x_2 are independent, and each of them takes two possible values, we can conclude that $x = (x_1, x_2)$ can take four possible values $(\underline{x}_1, \underline{x}_2)$, $(\underline{x}_1, \overline{x}_2)$, $(\overline{x}_1, \underline{x}_2)$, and $(\overline{x}_1, \overline{x}_2)$, and the probability of each of these values is equal to the product of the probabilities corresponding to x_1 and x_2. For this distribution, $E[\min(x_1, x_2)]$ is exactly the expression from the formulation of the theorem. Theorem 12 is proven.

Theorem 14. *For minimum $y = \min(x_1, x_2)$, when we have no information about the correlation between x_1 and x_2, we have $\overline{E} = \min(E_1, E_2)$,*

$$\underline{E} = \max(p_1 + p_2 - 1, 0) \cdot \min(\overline{x}_1, \overline{x}_2) + \min(p_1, 1 - p_2) \cdot \min(\overline{x}_1, \underline{x}_2) +$$
$$\min(1 - p_1, p_2) \cdot \min(\underline{x}_1, \overline{x}_2) + \max(1 - p_1 - p_2, 0) \cdot \min(\underline{x}_1, \underline{x}_2).$$

Theorem 15. *For maximum $y = \max(x_1, x_2)$, when we have no information about the correlation between x_1 and x_2, we have $\underline{E} = \max(E_1, E_2)$ and*

$$\overline{E} = \min(p_1, p_2) \cdot \max(\overline{x}_1, \overline{x}_2) + \max(p_1 - p_2, 0) \cdot \max(\overline{x}_1, \underline{x}_2) +$$
$$\max(p_2 - p_1, 0) \cdot \max(\underline{x}_1, \overline{x}_2) + \min(1 - p_1, 1 - p_2) \cdot \max(\underline{x}_1, \underline{x}_2).$$

Proof. We will prove Theorem 14; the proof of Theorem 15 is similar. Similarly to the proof of Theorem 12, we can conclude that $\min(E_1, E_2)$ is the attainable upper bound for $E[\min(x_1, x_2)]$. Due to convexity, to find the lower bound for $E[\min(x_1, x_2)]$, it is sufficient to consider distributions located at the four corners of the box $\mathbf{x}_1 \times \mathbf{x}_2$. Similar to the proof of Theorem 8, we conclude that such distribution can be characterized by a single parameter. Optimizing with respect to this parameter, we get the desired formula for $\underline{E}$. The theorem is proven.

Similar formulas can be produced for the cases when there is a strong correlation between x_i: namely, when x_1 is (non-strictly) increasing or decreasing in x_2.

From Elementary Arithmetic Operations to General Algorithms When we have a complex algorithm f, then a step-by-step approach leads to excess width. How can we find the actual range of $E = E[y]$?

At first glance, the exact formulation of this problem requires that we use infinitely many variables, because we must describe all possible probability distributions on the box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$ (or, in the independent case, all possible tuples consisting of distributions on all n intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$). It turns out, however, that we can reformulate these problems in equivalent forms that require only finitely many variables:

Theorem 16. *For a general continuous function $f(x_1, \ldots, x_n)$, $\underline{E}$ is a solution to the following optimization problem: $\sum_{j=0}^{n} p^{(j)} \cdot f(x_1^{(j)}, \ldots, x_n^{(j)}) \to \min$ under the conditions*

$$\sum_{k=0}^{n} p^{(k)} = 1; \quad p^{(j)} \geq 0; \quad \underline{x}_i \leq x_i^{(j)} \leq \overline{x}_i; \quad \underline{E}_i \leq \sum_{j=0}^{n} p^{(j)} \cdot x_i^{(j)} \leq \overline{E}_i \quad \text{(for all } i, j),$$

and $\overline{E}$ *is a solution to* $\sum_{j=0}^{n} p^{(j)} \cdot f(x_1^{(j)}, \ldots, x_n^{(j)}) \to \max$ *under the same constraints.*

Proof. In terms of the unknown probabilities $p^{(j)}$, we are minimizing a linear function under linear constraints (equalities and inequalities). Geometrically, the set of all points that satisfy several linear constraints is a polytope. It is well known that to find the minimum of a linear function on a polytope, it is sufficient to consider its vertices (this idea is behind linear programming). In algebraic terms, a vertex can be characterized by the fact that for N variables, N of the original constrains are equalities. Thus, in our case, all but n probabilities $p^{(j)}$ must be equal to 0. The theorem is proven.

6 Open Problems

So far, we have provided explicit formulas for the elementary arithmetic operations $f(x_1, \ldots, x_n)$ for the case when we know the first order moments. What if, in addition to that, we have some information about second order (and/or higher order) moments of x_i? What will we be then able to conclude about the moments of y? Partial answers to this question are given in [38, 60, 67]; it is desirable to find a general answer.

Similarly to Theorem 16, we can reduce the corresponding problems to the constraint optimization problems with finitely many variables. For example, when, in addition to intervals $\mathbf{E}_i$ that contain the first moments $E[x_i]$, we know the intervals $\mathbf{E}_{ik}$ that contain the second moments $E[x_i \cdot x_k]$, then the corresponding bounds $\underline{E}$ and $\overline{E}$ on $E[y]$ can be computed by solving the problems $\sum_{j=0}^{N} p^{(j)} \cdot f(x_1^{(j)}, \ldots, x_n^{(j)}) \to \min(\max)$ under the conditions

$$\sum_{j=0}^{N} p^{(j)} = 1; \quad p^{(j)} \geq 0; \quad \underline{x}_i \leq x_i^{(j)} \leq \overline{x}_i; \quad \underline{E}_i \leq \sum_{j=0}^{n} p^{(j)} \cdot x_i^{(j)} \leq \overline{E}_i;$$

$$\underline{E}_{ik} \leq \sum_{j=0}^{n} p^{(j)} \cdot x_i^{(j)} \cdot x_k^{(j)} \leq \overline{E}_{ik},$$

where $N = n(n+1)/2$.

It is desirable to find explicit analytical expressions for these bounds, at least for the case when $n = 2$ and $f(x_1, \ldots, x_n)$ is an elementary arithmetic operation.

7 Fast Quantum Algorithms for Handling Probabilistic and Interval Uncertainty

As computers become faster, quantum effects must be more and more taken into consideration. According to Moore's law, computer speed doubles every 18 months. One of the main limitations to further speedup is the computer size: every communication is limited by the speed of light c, so, e.g., a computer of a 1 ft size is bounded to have a computation speed 1 ft/c – which corresponds to 1 GHz. To make faster computers, we must thus decrease the size of computer elements. As this size reaches molecular size, must take into consideration quantum effects.

Quantum effects add to noise, but they can also help. Quantum effects, with their inevitably probabilistic behavior, add to noise. However, it turns out that some (intuitively counter-intuitive) quantum effects can be used to drastically speed up computations (in spite of quantum noise).

For example, without using quantum effects, we need – in the worst case – at least N computational steps to search for a desired element in an unsorted list of size N. A quantum computing algorithm proposed by Grover (see, e.g., [16, 17, 50]) can find this element much faster – in $O(\sqrt{N})$ time.

Several other quantum algorithms have been proposed.

What we are planning to do. How can this be of use to interval data processing community? In many application areas ranging from geosciences to bioinformatics to large-scale simulations of complex systems, data processing algorithms require a lot of time to run even with the exact input data. As a result, very little is currently done to analyze the effect of inevitable uncertainty of input data on the results of data processing.

It is desirable to analyze how different types of uncertainty–probabilistic, interval– influence the results of data processing. In this paper, we discuss how quantum algorithms such as Grover's quantum search can be used to speed up this analysis – and thus, make it possible.

We also explain that there is no need to wait until a full-blown quantum computer appears, with all necessary quantum bits ("qubits"): even without all necessary qubits, we can still get some speedup, a speedup that gets better and better as we add more qubits to the quantum computer.

Grover's algorithm for quantum search. We have already mentioned Grover's algorithm that, given:

- a database $a_1, \ldots, a_N$ with N entries,
- a property P (i.e., an algorithm that checks whether P is true), and
- an allowable error probability δ,

returns, with probability $\geq 1-\delta$, either the element a_i that satisfies the property P or the message that there is no such element in the database.

This algorithm requires $c \cdot \sqrt{N}$ steps (= calls to P), where the factor c depends on δ (the smaller δ we want, the larger c we must take).

General comment about quantum algorithms. For our applications, it is important to know that for Grover's algorithm (and for all the other quantum algorithms that we will describe and use), the entries a_i do not need to be all physically given, it is sufficient to have a procedure that, given i, produces a_i.

- If all the entries are physically given, then this procedure simply consists of fetching the i-th entry from the database.
- However, it is quite possible that the entries are given implicitly, e.g., a_i can be given as the value of a known function at i-th grid point; we have this function given as a program, so, when we need a_i, we apply this function to i-th grid point.

Algorithm for quantum counting. Brassard et al. used the ideas behind Grover's algorithm to produce a new quantum algorithm for *quantum counting*; see, e.g., [4, 50]. Their algorithm, given:

- a database $a_1, \ldots, a_N$ with N entries,
- a property P (i.e., an algorithm that checks whether P is true), and
- an allowable error probability δ,

returns an approximation $\widetilde{t}$ to the total number t of entries a_i that satisfy the property P.

This algorithm contains a parameter M that determines how accurate the estimates are. The accuracy of this estimate is characterized by the inequality

$$\left|\widetilde{t} - t\right| \le \frac{2\pi}{M} \cdot \sqrt{t} + \frac{\pi^2}{M^2} \tag{1}$$

that is true with probability $\ge 1 - \delta$.

This algorithm requires $c \cdot M \cdot \sqrt{N}$ steps (= calls to P), where the factor c depends on δ (the smaller δ we want, the larger c we must take).

In particular, to get the exact value t, we must attain accuracy $\left|\widetilde{t} - t\right| \le 1$, for which we need $M \approx \sqrt{N}$. In this case, the algorithm requires $O(\sqrt{t \cdot N})$ steps.

Quantum algorithms for finding the minimum. Dürr et al. used Grover's algorithm to produce a new quantum algorithm for *minimization*; see, e.g., [7, 50]. Their algorithm applied to the database whose entries belong to the set with a defined order (e.g., are numbers). This algorithm, given:

- a database $a_1, \ldots, a_N$ with N entries, and
- an allowable error probability δ,

returns the index i of the smallest entry a_i, with probability of error $\le \delta$.

This algorithm requires $c \cdot \sqrt{N}$ steps (= calls to P), where the factor c depends on δ (the smaller δ we want, the larger c we must take).

Main idea behind quantum computing of the minimum. The main idea behind the above algorithm can be illustrated on the example when all the entries a_i are integers. The algorithm requires that we know, e.g., a number M such that all the entries belong to the interval $[-M, M]$. For every value m between $-M$ and M, we can use Grover's algorithm to check whether there is an entry a_i for which $a_i < m$.

- If such an entry exists, then $m_0 \stackrel{\text{def}}{=} \min(a_i) < m$;
- otherwise $m_0 \ge m$.

Thus, for every m, we can check, in $O(\sqrt{N})$ steps, whether $m_0 < m$.

We can therefore apply bisection to narrow down the interval containing the desired until it narrows down to a single integer.

- We start with an interval $[\underline{M}, \overline{M}] = [-M, M]$.

- At each iteration, we pick a midpoint

$$m = \frac{\underline{M} + \overline{M}}{2}, \tag{2}$$

  and check whether $m_0 < M_0$.

  - If $m_0 < M_0$, this means that $m_0 \in [\underline{M}, M_0]$;
  - otherwise, $m_0 \in [M_0, \overline{M}]$.

  In both cases, we get a half-size interval containing m_0.

- After $\log_2(2M)$ iterations, this interval becomes so narrow that it can only contain one integer – which is m_0.

Thus, in $\log_2(M) \cdot O(\sqrt{N})$ steps, we can compute the desired minimum.

Quantum algorithm for computing the mean. The above algorithms can be used to compute the average of several numbers, and, in general, the mean of a given random variable. The first such algorithm was proposed by Grover in [18]; for further developments, see, e.g., [21, 49, 53].

The traditional Monte-Carlo method for computing the mean consists of picking M random values and averaging them. It is a well known fact [59, 68], that the accuracy of this method is $\sim 1/\sqrt{M}$, so, to achieve the given accuracy ε, we need $M \approx \varepsilon^{-2}$ iterations. Another way to compute the average of n given numbers is to add them up and divide by n, which requires n steps, Thus:

- when $n < \varepsilon^{-2}$, it is faster to add all the values;
- otherwise, it is better to use the Monte-Carlo method.

Grover's quantum analog of the Monte-Carlo method attains accuracy $\sim 1/M$ after M iterations; thus, for a given accuracy ε, we only need $M \approx \varepsilon^{-1}$ steps.

Similarly to the traditional Monte-Carlo methods, this quantum algorithm can compute multi-dimensional integrals $\int \ldots \int f(x_1, \ldots, x_n)\, dx_1 \ldots dx_n$: indeed, if we assume that the vector $(x_1, \ldots, x_n)$ is uniformly distributed over the corresponding domain, then this integral is proportional to the average value of $f(x_1, \ldots, x_n)$.

Quantum algorithms for probabilistic analysis. In the probabilistic case, the problem of describing the influence of the input uncertainty on the result of data processing takes the following form (see, e.g., [59, 68]). Given:

- the data processing algorithm $f(x_1, \ldots, x_n)$ that transforms any n input values $x_1, \ldots, x_n$ into the result of $y = f(x_1, \ldots, x_n)$ of data processing, and
- the mean values $\widetilde{x}_i$ and standard deviations σ_i of the inputs,

compute the standard deviation σ of the result y of data processing.

This standard deviation can be described as a mean (= mathematical expectation) of the square $(y - \widetilde{y})^2$, where

$$\widetilde{y} \stackrel{\text{def}}{=} f(\widetilde{x}_1, \ldots, \widetilde{x}_n), \quad y \stackrel{\text{def}}{=} f(x_1, \ldots, x_n), \tag{3}$$

and each x_i is normally distributed with mean $\widetilde{x}_i$ and standard deviation σ_i. Traditional Monte-Carlo algorithm requires $\sim 1/\varepsilon^2$ iterations to compute this average; thus, for accuracy 20%, we need 25 iterations; see, e.g., [65].

The quantum Monte-Carlo algorithm to compute this mean with accuracy ε in $\sim 1/\varepsilon$ iterations; so, for accuracy 20%, we only need 5 iterations. Since computing f may take a long time, this drastic (5 times) speed-up may be essential.

Quantum algorithms for interval computations: problem. In interval computations (see, e.g., [25, 26, 46]), the main objective is as follows. Given:

- intervals $[\underline{x}_i, \overline{x}_i]$ of possible values of the inputs $x_1, \ldots, x_n$, and
- the data processing algorithm $f(x_1, \ldots, x_n)$ that transforms any n input values $x_1, \ldots, x_n$ into the result of $y = f(x_1, \ldots, x_n)$ of data processing,

compute the exact range $[\underline{y}, \overline{y}]$ of possible values of y.

We can describe each interval in a more traditional form

$$[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i], \tag{4}$$

where $\widetilde{x}_i$ is the interval's midpoint, and Δ_i is its half-width. The resulting range can also be described as $[\widetilde{y} - \Delta, \widetilde{y} + \Delta]$, where $\widetilde{y}$ is determined by (3), and Δ is the desired largest possible difference $|y - \widetilde{y}|$.

Quantum algorithms for interval computations: case of relatively small errors. When the input errors are relatively small, we can linearize the function f around the midpoints $\widetilde{x}_i$. In this case, Cauchy distributions turn out to be useful, with probability density

$$\rho(x) \sim \frac{1}{1 + \dfrac{(x-a)^2}{\Delta^2}}. \tag{5}$$

It is known [65] that if we take x_i distributed according to Cauchy distribution with a center $a = \widetilde{x}_i$ and the width parameter Δ_i, then the difference $y - \widetilde{y}$ between the quantities (3) is also Cauchy distributed, with the width parameter equal to the desired value Δ.

For Cauchy distribution, the standard deviation is infinite, so we cannot literally apply the idea that worked in the probabilistic case. However, if we apply a function $g(x)$ (e.g., arctan) that reduces the entire real line to an interval, then the expected value of $g\left((y - \widetilde{y})^2\right)$ – that depends only on Δ – can be computed by the quantum Monte-Carlo algorithm; from this value, we can reconstruct Δ.

So, in this case, quantum techniques also speed up computations.

Quantum algorithms for interval computations: general case. Known results about the computational complexity of interval computations (see, e.g., [33]) state that in the general case, when the input errors are not necessarily small and the function f may be complex, this problem is NP-hard. This, crudely speaking, means that in the worst case, we cannot find the exact range for y faster than by using some version of exhaustive search of all appropriate grid points.

The problem is not in exactness: it is also known that the problem of computing the range with a given approximation accuracy ε is also NP-hard.

How can we actually compute this range? We can find, e.g., $\underline{y}$ with a given accuracy δ as follows. The function f is continuous; hence, for a given ε, there exists an δ such that the $\leq \delta$-difference in x_i leads to $\leq \delta$ change in y. Thus, within a given accuracy ε, it is sufficient to consider a grid with step δ, and take the smallest of all the values of f on this grid as $\underline{y}$.

If the linear size of the domain is D, then, in this grid, we have D/δ values for each of the variables, hence, the total of $(D/\delta)^n$ points.

In non-quantum computations, to compute the minimum, we need to check every points from this grid, so we must use $N = (D/\delta)^n$ calls to f. The quantum algorithm for computing minimum enables to use only $\sqrt{N} = (D/\delta)^{n/2}$ calls.

Thus, quantum algorithms can double the dimension of the problem for which we are able to compute the desired uncertainty.

Quantum algorithms for the case when we have several different types of uncertainty. How can we extend the above results to the case when we have several different types of uncertainty? In this section, we present preliminary result about the case when we have both probabilistic and interval uncertainty.

When we have n measurement results $x_1, \ldots, x_n$, traditional statistical approach usually starts with computing their sample average

$$E = \frac{x_1 + \ldots + x_n}{n} \tag{6}$$

and their sample variance

$$V = \frac{(x_1 - E)^2 + \ldots + (x_n - E)^2}{n} \tag{7}$$

(or, equivalently, the sample standard deviation $\sigma = \sqrt{V}$); see, e.g., [59]. If we know the exact values of x_i, then these formulas require linear computation time $O(n)$.

As we have mentioned, in many practical situations, we only have intervals $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ of possible values of x_i. As a result, the sets of possible values of E and V are also intervals.

The function E is monotonic in each x_i, so the range $[\underline{E}, \overline{E}]$ for E can be easily computed:

$$\underline{E} = \frac{\underline{x}_1 + \ldots + \underline{x}_n}{n}; \quad \overline{E} = \frac{\overline{x}_1 + \ldots + \overline{x}_n}{n}. \tag{8}$$

In [9, 10], we have shown that the problem of computing the range $[\underline{V}, \overline{V}]$ is, in general, NP-hard (even when we are interested in computing this range with a given accuracy); we have also described a quadratic-time $O(n^2)$ algorithm $\underline{\mathcal{A}}$ for computing $\underline{V}$ and a a quadratic-time algorithm $\overline{\mathcal{A}}$ that computes $\overline{V}$ for all the cases in which, for some integer C, no more than C "narrowed" intervals $[\widetilde{x}_i - \Delta_i/n, \widetilde{x}_i + \Delta_i/n]$ can have a common intersection.

Let us first show that by using Monte-Carlo simulations, we can compute $\underline{V}$ with given accuracy in time $O(n \cdot \log_2(n)) \ll O(n^2)$; to be more precise, we need time $O(n \cdot \log_2(n))$ time to sort $2n$ values and then $O(n)$ steps to complete the computations.

Indeed, the algorithm $\underline{\mathcal{A}}$ from [9, 10] is as follows:

- First, we sort all $2n$ values $\underline{x}_i$, $\overline{x}_i$ into a sequence $x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(2n)}$.
- Second, we compute $\underline{E}$ and $\overline{E}$ and select all zones $[x_{(k)}, x_{(k+1)}]$ that intersect with $[\underline{E}, \overline{E}]$.

- For each of the selected small zones $[x_{(k)}, x_{(k+1)}]$, we compute the ratio $r_k = S_k/N_k$, where

$$S_k \stackrel{\text{def}}{=} \sum_{i:\underline{x}_i \geq x_{(k+1)}} \underline{x}_i + \sum_{j:\overline{x}_j \leq x_{(k)}} \overline{x}_j, \tag{9}$$

  and N_k is the total number of such is and js. If $r_k \in [x_{(k)}, x_{(k+1)}]$, then we compute V_k' as

$$\frac{1}{n} \cdot \left(\sum_{i:\underline{x}_i \geq x_{(k+1)}} (\underline{x}_i - r_k)^2 + \sum_{j:\overline{x}_j \leq x_{(k)}} (\overline{x}_j - r_k)^2 \right).$$

  If $N_k = 0$, we take $V_k' \stackrel{\text{def}}{=} 0$.

- Finally, we return the smallest of the values V_k' as $\underline{V}$.

For each k, the value r_k is a mean, so, by using Monte-Carlo methods, we can compute it in time that does not depend on n at all; similarly, we can compute V_k' in constant time. The only remaining step is to compute the smallest of $\leq 2n$ values V_k'; this requires $O(n)$ steps.

If quantum computing is available, then we can compute the minimum in $O(\sqrt{n})$ steps; thus, we only need $O(\sqrt{n})$ steps after sorting.

Similarly, the algorithm $\overline{\mathcal{A}}$ is as follows:

- First, we sort all $2n$ endpoints of the narrowed intervals $\widetilde{x}_i - \Delta_i/n$ and $\widetilde{x}_i + \Delta_i/n$ into a sequence $x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(2n)}$. This enables us to divide the real line into $2n+1$ zones $[x_{(k)}, x_{(k+1)}]$, where we denoted $x_{(0)} \stackrel{\text{def}}{=} -\infty$ and $x_{(2n+1)} \stackrel{\text{def}}{=} +\infty$.

- Second, we compute $\underline{E}$ and $\overline{E}$ and select all zones $[x_{(k)}, x_{(k+1)}]$ that intersect with $[\underline{E}, \overline{E}]$.

- For each of remaining zones $[x_{(k)}, x_{(k+1)}]$, for each i from 1 to n, we pick the following value of x_i:

  - if $x_{(k+1)} < \widetilde{x}_i - \Delta_i/n$, then we pick $x_i = \overline{x}_i$;
  - if $x_{(k)} > \widetilde{x}_i + \Delta_i/n$, then we pick $x_i = \underline{x}_i$;
  - for all other i, we consider both possible values $x_i = \overline{x}_i$ and $x_i = \underline{x}_i$.

  As a result, we get one or several sequences of x_i. For each of these sequences, we check whether the average E of the selected values $x_1, \ldots, x_n$ is indeed within the corresponding zone, and if it is, we compute the sample variance by using the formula (7).

- Finally, we return the largest of the computed sample variances as $\overline{V}$.

It is shown that we end up with $\leq 2^C \cdot 2n = O(n)$ sample variances.

Here also, computing E and V can be done in constant time, and selecting the largest of $O(n)$ variances requires linear time $O(n)$ for non-quantum computations and $O(\sqrt{n})$ time for quantum computing.

Can quantum computers be still useful when there are not yet enough qubits? In view of the great potential for computation speedup, engineers and physicists are actively working on the design of actual quantum computers. There already exist working prototypes: e.g., a several mile long communication system, with simple quantum computers used for encoding and decoding, is at government disposal. Microsoft and IBM actively work on designing quantum computers. However, at present, these computers can only solve trivial instances of the above problems, instances that have already been efficiently solved by non-quantum computers. Main reason: the existing quantum computers have only a few qubits, while known quantum algorithms require a lot of qubits. For example, Grover's algorithm requires a register with $q = \log(N)$ qubits for a search in a database of n elements.

Of course, while we only have 2 or 3 or 4 qubits, we cannot do much. However, due to the active research and development in quantum computer hardware, we will (hopefully) have computers with larger number of qubits reasonably soon.

A *natural question* is: while we are still waiting for the qubit register size that is necessary to implement the existing quantum computing algorithms (and thus, to achieve the theoretically possible speedup), can we somehow utilize the registers of smaller size to achieve a partial speed up?

In this section, we start answering this question by showing the following: for quantum search, even when we do not have enough qubits, we can still get a partial speedup; for details, see [40]. The fact that we do get a partial speedup for quantum search makes us hope that even when we do not have all the qubits, we can still get a partial speedup for other quantum computing algorithms as well.

Let us assume that we are interested in searching in an unsorted database of n elements, and that instead of all $\log(N)$ qubits that are necessary for Grover's algorithm, we only have, say 90% or 50% of them. To be more precise, we only have a register consisting of $r = \alpha \cdot \log(N)$ qubits, where $0 < \alpha < 1$. How can we use this register to speed up the search?

Grover's algorithm enables us to use a register with r qubits to search in a database of $M = 2^r$ elements in time $C \cdot \sqrt{M}$. For our available register, $r = \alpha \cdot \log(N)$, hence $M = 2^r = N^\alpha$, so we can use Grover's algorithm with this qubit register to search in a database of size N^α in time $C \cdot \sqrt{M} = C \cdot N^{\alpha/2}$.

To search in the original database of size N, we can do the following:

- divide this original database into $N^{1-\alpha}$ pieces of size N^α; and then
- consequently apply Grover's algorithm with a given qubit register to look for the desired element in each piece.

Searching each piece requires $C \cdot N^{\alpha/2}$ steps, so the sequential search in all $N^{1-\alpha}$ pieces requires time $N^{1-\alpha} \cdot (C \cdot N^{\alpha/2}) = C \cdot N^{1-\alpha/2}$. Since $\alpha > 0$, we get a speedup.

When α tends to 0, the computation time tends to $C \cdot N$, i.e., to the time of non-quantum search; when α tends to 1, the computation time tends to $C \cdot N^{1/2}$, i.e., to the time of quantum search.

8 Does "NP-Hard" Really Mean "Intractable"?

Introduction. Most of the computational problems related to interval computations are, in general, NP-hard. Most computer scientists believe that NP-hard problem are really computationally intractable. This belief is well justified for traditional computers, but there are non-traditional

physical and engineering ideas that may make NP-hard problem easily solvable. Let us briefly overview these ideas.

Within Newtonian physics, NP-hardness does seem to mean "intractable". The common belief that NP-hard means intractable is based on the abilities of the physical processes that are used in the existing computers; it has been proven that is we only use processes from Newtonian physics, then we do not add additional ability to the computational devices (for exact formulations and proofs, see, e.g., Gandy [12, 13]).

Within traditional (Newtonian) physical and engineering solutions, NP-hard seems to indeed mean "intractable". Indeed, the existing computations schemes describe (more or less accurately) the ability of the modern computers. The only thing that is missing from the standard algorithms is *randomness*, i.e., the ability to input *truly random* data and use them in computations. In the language of theory of computation, the outside source of data is called an *oracle*. As early as 1981, Bennet *et al.* have shown [2] that if we allow a random sequence as an oracle, and correspondingly reformulate the definitions of the classes P and NP, then we can *prove* that P$\neq$NP [2].

What if we use non-traditional physical and engineering ideas in computer design? Since we seem not to be able to avoid the unrealistic exponential time with traditional, Newtonian-physics-based computers, a question naturally appears: what if in the future, we will find *non-Newtonian* processes; will then NP-hard problems still be intractable? This question was first formulated by G. Kreisel [36].

Traditional computers use discrete-oriented deterministic processes in normal space and time. In reality, physical processes are (1) continuous, (2) non-deterministic (as described by *quantum mechanics*), and (3) they occur in non-traditional (*curved*) space-time. So, to describe how using additional physical processes will help in computations, we must consider how these three factors (adding non-determinism and taking curvature into consideration) change our computational abilities.

Non-Newtonian processes of first type: Use of physical fields. For a physical field, the value $f(\vec{x}, t)$ of the field f in a future moment of time t can be expressed in terms of the current state of this field $f(\vec{x}, 0)$ by an explicit integral formula. This formula is usually computable on existing computers, and therefore, the evolution described by the fields is *recursive* (relevant theorems are proved, e.g., in Pour-El *et al.* [58]).

In some cases, however, $f(\vec{x}, t)$ is described as an integral in terms of the function $f(\vec{x}, t)$ and its spatial derivatives. So, if we start with a function $f(\vec{x}, 0)$ that is recursive, but whose (spatial) derivatives are not recursive, we may end up with a *non-recursive value* $f(\vec{x}, t)$. This was shown by Pour-El *et al.* in [57] (see also Beeson [1], Ch. 15, and Pour-El *et al.* [58]). This result generalizes a theorem proved by Aberth in 1971 and rediscovered in Pour-El *et al.* [56].

Comment. This result does not necessarily mean that we have found a way to compute a function that is not computable on a normal computer (see, e.g., Kreisel [37]), because for that, we would need to find a way to implement the initial conditions with a non-recursive derivative. A more definite possibility of solving NP-hard problem fast comes from the other two aspects of physical processes.

Non-Newtonian processes of second type: Quantum processes (adding non-determinism). We have already mentioned that quantum computing can solve several useful

problems–like factoring integers–faster than non-quantum ones; see, e.g., [50]. It may be possible to use them for solving NP-hard problems. Several such hypothetic schemes–using quantum field theory–have been proposed by Freedman, Kitaev. etc.

We propose to use one more phenomenon: namely, some potentially observable dependencies between physical quantities become not only non-smooth but even discontinuous; these cases have been summarized in a recent monograph [15]. From the computational viewpoint, this seems to open the doors to the possibility of checking whether a given real number is equal to 0 or not, something that the halting problem explicitly prohibits us from doing for normal (non-quantum) computing. We are therefore planning to investigate the possibility of using this new opportunity in actual computing. Our preliminary results are presented in [20].

Non-Newtonian processes of third type: Using curved space-time for computations. If we allow heavily curved space (e.g., semi-closed *black holes*, we can get the results faster if we stay in the area where the curvature is strong and time goes slower, and let the computations be done outside (see, e.g., Morgenstein *et al.* [30, 48]); then, we will even be able to compute NP-hard problems in polynomial time.

A similar speed-up can be obtained if we assume that our space-time is hyperbolic [24, 30, 48].

Previously described non-Newtonian processes relate to well-recognized physics, but there are other physical theories that describe possible but not universally accepted physics. Namely, several physical theories has led to the appearance of closed timelike curves, the possibility to go back in time. Suffice it to say that one of the main ideas which helped R. Feynman to develop a modern version of quantum electrodynamics was the idea of positrons as electrons going back in time [11].

Until late 1980s, these possibilities were largely dismissed by mainstream physics as mathematical artifacts which cannot lead to actual going back in time. Only when Kip Thorne, the world's leading astrophysicist, published several papers on acausal solutions in *Physical Reviews*, the topic became more mainstream.

Acausal anomalies were discovered in solutions of general relativity (around a massive fast rotating cylinder), in string theory, in inflation-theory related cosmological models, etc.; see, e.g., [64] and references therein.

The main obstacle to accepting acausal phenomena used to be paradoxes associated with time travel. These paradoxes can be illustrated on a commonsense example of a "father paradox": a time traveler can go to the past and kill his father before he himself was conceived, thus creating a paradox. The accepted solution to the acausal paradoxes can also be illustrated on the "father paradox" example: since the time traveler was born, this means that some unexpected event prevented him from killing his father. Maybe a policeman stopped him, maybe his gun malfunctioned. Even is the time traveler takes care of all such probably events, there are always events with small probability – like a meteorite falling on the traveler's head – which cannot be all avoided. Thus, all we will achieve if we try to implement a paradox is that some event with a very low probability will occur.

There are several ways to use acausal processes in computing. The trivial way is to let a computer run a long program for whatever it takes and then send the results back in time. A less trivial way of saving time is similar to quantum "computing without computing" – speeding up without actually using time machines. This method was originally proposed in [29]; see also [28, 42, 47]. This method is related to the above solution to the father paradox. Indeed, to solve, e.g., a propositional satisfiability problem with n variables, we generate n random bits and check whether they satisfy a given formula; if not, we launch a time machine that is set up to implement a low-probability event (with some probability $p_0 \ll 1$). Nature has two choices: either it generates

n variables which satisfy the given formula (probability of this is 2^{-n}), or the time machine is used which leads to an event with a probability p_0. If $2^{-n} \gg p_0$, then statistically, the first event is much more probable, and so, the solution to the satisfiability problem will actually be generated without the actual use of a time machine.

Yet another possibility: Gell-Mann's approach to complex systems. In several papers and in his book [14], a Nobelist physicist M. Gell-Mann suggests that our difficulties in describing the dynamics of complex systems can be resolved if we assume that the true physical equations actually explicitly contain Kolmogorov complexity of the described system. In [34], we show that a natural physical approach indeed leads to new equations which are equivalent to an explicit incorporation of Kolmogorov complexity. Since Kolmogorov complexity is not computable (see, e.g., [39]), the possibility described by Gell-Mann's hypothesis can be used to speed up computations.

Acknowledgments.

This work was supported in part by NASA under cooperative agreement NCC5-209, by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-1-0365, by NSF grants EAR-0112968 and EAR-0225670, by the Army Research Laboratories grant DATM-05-02-C-0046, and by IEEE/ACM SC2001 and SC2002 Minority Serving Institutions Participation Grants.

This work was partly supported by a travel grant from the conference organizers.

References

[1] Beeson M. J. (1985). "Foundations of constructive mathematics", Springer-Verlag, N.Y.

[2] Bennet G. G., Gill J. (1981). "Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co-NP}^A$ with probability 1", *SIAM Journal of Computer Science, vol. 10*, 96–113.

[3] Blum L., Shub M., Smale S. (1989). "On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions, and universal machines", *Bull. Amer. Math. Soc., vol. 21*, 1–46.

[4] Brassard G., Hoyer P., Tapp A. (1998). Quantum counting. In: *Proc. 25th ICALP*, Lecture Notes in Computer Science, Vol. 1443, Springer, Berlin, 820–831.

[5] Cormen, Th. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001) "Introduction to Algorithms", MIT Press, Cambridge, MA.

[6] Davenport J. H., Heintz J. (1988). "Real quantifier elimination is doubly exponential", *Journal of Symbolic Computations, vol. 5(1/2)*, 29–35.

[7] Dürr C., Hoyer P. (1996). "A quantum algorithm for finding the minimum"; LANL arXiv:quant-ph/9607014

[8] Emde Boas P. van (1990), "Machine models and simulations", In: J. van Leeuwen (ed.), *Handbook on Theoretical Computer Science*, Elsevier Science, N.Y., 1–63.

[9] Ferson S., Ginzburg L., Kreinovich V., Longpré L., and Aviles M. (2002), "Computing Variance for Interval Data is NP-Hard", *ACM SIGACT News, vol. 33*, 108–118.

[10] Ferson S., Ginzburg L., Kreinovich V., and Lopez J. (2002), "Absolute Bounds on the Mean of Sum, Product, etc.: A Probabilistic Extension of Interval Arithmetic", *Extended Abstracts of the 2002 SIAM Workshop on Validated Computing*, Toronto, Canada, May 23–25, 70–72.

[11] Feynman R. P. (1949). "The theory of positrons", *Physical Review, vol. 76*, 749–759.

[12] Gandy, R. (1980). "Church's thesis and principles for mechanisms", In: J. Barwise, H. J. Keisler, and K. Kunen, *The Kleene Symposium*, North Holland, Amsterdam, 123–148.

[13] Gandy, R. (1988). "The confluence of ideas in 1936", In: R. Herken (ed.) *The universal Turing machine: a half-century survey*, Kammerer & Unverzagt, Hamburg.

[14] Gell-Mann M. (1994). "The Quark and the Jaguar", Freeman, N.Y.

[15] Grib A. A., Rodriguez W. .A. Jr. (1999). "Nonlocality in quantum physics", Plenum, N.Y.

[16] Grover L. (1996). A fast quantum mechanical algorithm for database search, *Proc. 28th ACM Symp. on Theory of Computing*, 212–219.

[17] Grover L. K. (1997). Quantum mechanics helps in searching for a needle in a haystack, *Phys. Rev. Lett., vol. 79*, 325–328.

[18] Grover L. (1998). A framework for fast quantum mechanical algorithms, *Phys. Rev. Lett., vol. 80*, 4329–4332.

[19] Hansen, E. (1997), "Sharpness in interval computations", *Reliable Computing, vol. 3*, 7–29.

[20] Harary F., Kreinovich V., Longpré L. (2001). "A new graph characteristic and its application to numerical computability", *Information Processing Letters, vol. 77*, 277–282.

[21] Heinrich S. (2002). Quantum summation with an application to integration, *J. Complexity, vol. 18(1)*, 1–50.

[22] Heinrich S., Novak E. (2000). Optimal summation and integration by deterministic, randomized, and quantum algorithms, In: F. Hickernell and H. Niederreiter (eds.), *Proc. 4th Int'l Conf. on Monte Carlo and Quasi-Monte Carlo Methods*, Hong Kong.

[23] Heinrich S., Novak E. (2003). On a problem in quantum summation, *J. Complexity* (to appear).

[24] Herrmann F., Margenstern M. (2003). "A universal cellular automaton in the hyperbolic plane", *Theoretical Computer Science, vol. 296*, No. 2, 327–364.

[25] Jaulin L., Keiffer M., Didrit O., and Walter E. (2001), "Applied Interval Analysis", Springer-Verlag, Berlin.

[26] Kearfott R. B. (1996), "Rigorous Global Search: Continuous Problems", Kluwer, Dordrecht.

[27] Kearfott R. B. and Kreinovich V., eds. (1996), "Applications of Interval Computations" (Pardalos. P. M., Hearn, D., "Applied Optimization", Vol. 3), Kluwer, Dordrecht.

[28] Koshelev M. (1998). "Maximum entropy and acausal processes: astrophysical applications and challenges", In: G. J. Erickson et al. (eds.), *Maximum Entropy and Bayesian Methods*, Kluwer, Dordrecht, 253–262.

[29] Kosheleva O. M., Kreinovich V. (1981). "What can physics give to constructive mathematics", In: *Mathematical Logic and Mathematical Linguistics*, Kalinin, Russia, 117–128 (in Russian).

[30] Kreinovich V. (1989). "On the possibility of using physical processes when solving hard problems", Leningrad Center for New Information Technology "Informatika", Technical Report, Leningrad (in Russian).

[31] Kreinovich V. (2000), "Beyond Interval Systems: What Is Feasible and What Is Algorithmically Solvable?", In: Pardalos P. M., ed., "Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems", Kluwer, Dordrecht, 364–379.

[32] Kreinovich V. (2003) "Probabilities, Intervals, What Next? Optimization Problems Related to Extension of Interval Computations to Situations with Partial Information about Probabilities", *Journal of Global Optimization* (to appear).

[33] Kreinovich V., Lakeyev A., Rohn J., and Kahl P. (1997), "Computational Complexity and Feasibility of Data Processing and Interval Computations" (Pardalos. P. M., Hearn, D., "Applied Optimization", Vol. 10), Kluwer, Dordrecht.

[34] Kreinovich V., Longpré L. (1998). "Why Kolmogorov Complexity in Physical Equations", *Int'l J. of Theor. Physics, vol. 37*, 2791–2801.

[35] Kreinovich V., Nguyen H. T., Ferson S., and Ginzburg L. (2002), "From Computation with Guaranteed Intervals to Computation with Confidence Intervals", *Proc. 21st Int'l Conf. of North American Fuzzy Information Processing Society NAFIPS'2002*, New Orleans, Louisiana, 418–422.

[36] Kreisel G. (1974). "A notion of mechanistic theory", *Synthese, vol. 29*, 11–26.

[37] Kreisel G. (1982). "A review of [56] and [57]", *Journal of Symbolic Logic, vol. 47*, 900–902.

[38] Kuznetsov V. P. (1991), "Interval Statistical Models", Radio i Svyaz, Moscow (in Russian).

[39] . Li M., Vitányi, P. M. B. (1997). "An Introduction to Kolmogorov Complexity and its Applications", Springer-Verlag, N.Y.

[40] Longpré L., Kreinovich V. (2003). Can Quantum Computers Be Useful When There Are Not Yet Enough Qubits?", Bull. European Association for Theoretical Computer Science (EATCS), vol. 79, 164–169.

[41] Martinez M., Longpré L., Kreinovich V., Starks S. A., Nguyen H. T. (2003). "Fast Quantum Algorithms for Handling Probabilistic, Interval, and Fuzzy Uncertainty", *Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society NAFIPS'2003*, Chicago, Illinois, July 24–26, 2003.

[42] Maslov S. Yu. (1987). "Theory Of Deductive Systems And Its Applications", MIT Press, Cambridge, MA.

[43] Meer K. (1994). "On the complexity of quadratic programming in real number models of computation", *Theoretical Computer Science, vol. 133*, 85–94.

[44] Meer K. (1994), "Real number computations: on the use of information", *J. Symbolic Computation, vol. 18*, 199–206.

[45] Meer K. (1995). "On the relations between discrete and continuous complexity theory", *Math. Log. Quarterly, vol. 41(2)*, 281–286.

[46] Moore R. E. (1979), "Methods and Applications of Interval Analysis", SIAM, Philadelphia.

[47] Moravec H. (1991). "Time travel and computing", Carnegie-Mellon Univ., CS Dept. Preprint.

[48] Morgenstein D., Kreinovich V. (1995). "Which algorithms are feasible and which are not depends on the geometry of space-time", *Geombinatorics, vol. 4*, No. 3, 80–97.

[49] Nayak A., Wu, F. (1999). The quantum query complexity of approximating the median and related statistics, *Proc. Symp. on Theory of Computing STOC'99*, 384–393.

[50] Nielsen M. A., Chuang I. L. (2000). *Quantum computation and quantum information*, Cambridge University Press, Cambridge, U.K.

[51] Nivlet P., Fournier F., and Royer J. (2001), "A new methodology to account for uncertainties in 4-D seismic interpretation", *Proceedings of the 71st Annual International Meeting of the Society of Exploratory Geophysics SEG'2001*, San Antonio, Texas, September 9–14, 1644–1647.

[52] Nivlet P., Fournier F., and Royer J. (2001), "Propagating interval uncertainties in supervised pattern recognition for reservoir characterization", *Proceedings of the 2001 Society of Petroleum Engineers Annual Conference SPE'2001*, New Orleans, Louisiana, September 30–October 3, paper SPE-71327.

[53] Novak E. (2001). Quantum Complexity of integration, *J. Complexity, vol. 17*, 2–16.

[54] Osegueda, R., Kreinovich, V., Potluri, L., and Aló R. (2002), "Non-Destructive Testing of Aerospace Structures: Granularity and Data Mining Approach", *Proceedings of FUZZ-IEEE'2002*, Honolulu, Hawaii, May 12–17, *Vol. 1*, 685–689.

[55] Pan V. (1997). "Solving a polynomial equation: some history and recent progress", *SIAM Review, vol. 39(2)*, 187–220.

[56] Pour-El M. B., Richards I. (1979). "A computable ordinary differential equation which possesses no computable solutions", *Annals of Mathematical Logic, vol. 17*, pp. 61–90.

[57] Pour-El M. B., Richards I. (1981). "The wave equation with computable initial data such that its unique solution is not computable", *Advances in Mathematics, vol. 39*, pp. 215–139.

[58] Pour-El M. B., Richards I. (1989). "Computability in analysis and physics", Springer-Verlag, N.Y.

[59] Rabinovich S. (1993), "Measurement Errors: Theory and Practice", American Institute of Physics, New York.

[60] Rowe, N. C. (1988), "Absolute bounds on the mean and standard deviation of transformed data for constant-sign-derivative transformations", *SIAM Journal of Scientific Statistical Computing, vol. 9*, 1098–1113.

[61] Shor P. (1994). "Algorithms for quantum computation: Discrete logarithms and factoring", *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, 124–134.

[62] Smale S. (1990). "Some remarks on the foundations of numerical analysis", *SIAM Review, vol. 32(2)*, 211–220.

[63] Tarski A. (1951). "A decision method for elementary algebra and geometry", 2nd ed., Berkeley and Los Angeles, 1951.

[64] Thorne K. S. (1994). "From black holes to time warps", W.W. Norton, N.Y.

[65] Trejo R., Kreinovich V. (2001). Error Estimations for Indirect Measurements: Randomized vs. Deterministic Algorithms. In: S. Rajasekaran et al. (eds.), *Handbook on Randomized Computing*, 673–729.

[66] Vavasis S. A. (1991), "Nonlinear Optimization: Complexity Issues", Oxford University Press, N.Y.

[67] Walley, P. (1991), "Statistical Reasoning with Imprecise Probabilities", Chapman and Hall, N.Y.

[68] Wadsworth H. M. Jr., ed. (1990). "Handbook of statistical methods for engineers and scientists", McGraw-Hill Publishing Co., N.Y.