9-2015

Combining Interval and Probabilistic Uncertainty: What Is Computable?

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

Andrzej Pownuk
The University of Texas at El Paso, ampownuk@utep.edu

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

Comments:
Technical Report: UTEP-CS-15-66
To appear in: Panos Pardalos, Anatoly Zhigljavsky, and Julius Zilinskas, *Advances in Stochastic and Deterministic Global Optimization*, Springer Verlag.

Combining Interval and Probabilistic Uncertainty: What Is Computable?

Vladik Kreinovich, Andrzej Pownuk, and Olga Kosheleva
Computational Science Program
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
vladik@utep.edu, ampownuk@utep.edu, olgak@utep.edu

Abstract

In many practical problems, we need to process measurement results. For example, we need such data processing to predict future values of physical quantities. In these computations, it is important to take into account that measurement results are never absolutely exact, that there is always measurement uncertainty, because of which the measurement results are, in general, somewhat different from the actual (unknown) values of the corresponding quantities. In some cases, all we know about measurement uncertainty is an upper bound; in this case, we have an *interval* uncertainty, meaning that all we know about the actual value is that is belongs to a certain interval. In other cases, we have some information – usually partial – about the corresponding probability distribution. New data processing challenges appear all the time; in many of these cases, it is important to come up with appropriate algorithms for taking uncertainty into account.

Before we concentrate our efforts on designing such algorithms, it is important to make sure that such an algorithm is possible in the first place, i.e., that the corresponding problem is algorithmically computable. In this paper, we analyze the computability of such uncertainty-related problems. It turns out that in a naive (straightforward) formulation, many such problems are not computable, but they become computable if we reformulate them in appropriate practice-related terms.

1 Formulation of the Problem

Need for data processing. In practice, we are often interested in a quantity y which is difficult to measure directly. Examples of such quantities are distance to a star, amount of oil in the well, or tomorrow's weather. This is important, since one of the main objectives of science is to predict future values of different quantities.

To estimate such quantities, we find easier-to-measure quantities $x_1, \ldots, x_n$ which related to y by a known dependence $y = f(x_1, \ldots, x_n)$. For example, to predict the future values of important quantities, we can use the known relations between the current and future values of different quantities.

Once such a relation is known, we measure the auxiliary x_i and use the measurement results $\widetilde{x}_i$ to compute an estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ for the desired quantity y. For example, to predict the future values of physical quantities, we use the results $\widetilde{x}_i$ of measuring the current values x_i of these (and related) physical quantities and the known relations $y = f(x_1, \ldots, x_n)$ between the current (x_i) and future (y) values of different quantities.

The corresponding estimation is what constitutes *data processing*.

Need to take uncertainty into account when processing data. The resulting estimates are never 100% accurate:

- measurements are never absolutely accurate,
- physical models used for predictions are usually only approximate, and
- sometimes (like in quantum physics) these models only predict the probabilities of different events.

It is desirable to take this uncertainty into account when processing data.

In some cases, we know all the related probabilities; in this case, we can potentially determine the values of all statistical characteristics of interest: mean, standard deviation, correlations, etc.

In most practical situations, however, we only have partial information about the corresponding probabilities. For example, for measurement uncertainties, often, the only information that we have about this uncertainty is the upper bound Δ on its absolute value; in this case, after we get a measurement result $\widetilde{X}$, the only information that we have about the actual (unknown) value of the corresponding quantity X is that it belongs to the interval $[\widetilde{X} - \Delta, \widetilde{X} + \Delta]$. We may know intervals containing the actual (unknown) cumulative distribution function, we may know bounds on moments, etc. In such situations of partial knowledge, for each statistical characteristic of interest, we can have several possible values. In such cases, we are interested in the interval of possible values of this characteristic, i.e., in the smallest and the largest possible values of this characteristic. In some cases, there are efficient algorithms for computing these intervals, in other cases, the corresponding general problem is known to be NP-hard or even not algorithmically computable; see, e.g., [3].

Studying computability – just like studying NP-hardness – is important, since it prevents us from vain attempts to solve the problem in too much generality, and helps us concentrate on doable cases. In view of this importance, this paper, we describe the most general related problems which are still algorithmically solvable.

2 What Is Computable: A Brief Reminder

What is computable: general idea. We are interested in processing uncertainty, i.e., in dealing with a difference between the exact models of physical reality and our approximate representation of this reality. In other words, we are interested in models of physical reality.

Why do we need mathematical models in the first place? One of our main objectives is to predict the results of different actions (or the result of not performing any action). Models enable us to predict these results without the need to actually perform these actions, thus often drastically decreasing potential costs. For example, it is theoretically possible to determine the stability limits of an airplane by applying different stresses to several copies of this airplane until each copy breaks, but, if we have an adequate computer-based model, it is cheaper and faster to simulate different stresses on this model without having to destroy actual airplane frames.

From this viewpoint, a model is computable if it has algorithms that allow us to make the corresponding predictions. Let us recall how this general idea can be applied to different mathematical objects.

What is computable: case of real numbers. In modeling, real numbers usually represent values of physical quantities. This is what real numbers were originally invented for – to describe quantities like length, weight, etc., this is still one of the main practical applications of real numbers.

The simplest thing that we can do with a physical quantity is measure its value. In line with the above general idea, we can say that a real number is computable if we can predict the results of measuring the corresponding quantity.

A measurement is practically never absolutely accurate, it only produces an approximation $\widetilde{x}$ to the actual (unknown) value x; see, e.g., [6]. In modern computer-based measuring instruments, such an approximate value $\widetilde{x}$ is usually a binary fraction, i.e., a rational number.

For every measuring instrument, we usually know the upper bound Δ on the absolute value of the corresponding measurement error $\Delta x \stackrel{\text{def}}{=} \widetilde{x} - x$: $|\Delta x| \le \Delta$. Indeed, without such a bound, the difference Δx could be arbitrary large, and so, we would not be able to make any conclusion about the actual value x; in other words, this would be a wild guess, not a measurement.

Once we know Δ, then, based on the measurement result $\widetilde{x}$, we can conclude that the actual value x is Δ-close to $\widetilde{x}$: $|x - \widetilde{x}| \le \Delta$. Thus, it is reasonable to say that a real number x is computable if for every given accuracy $\Delta > 0$, we can efficiently generate a rational number that approximates x with the given accuracy.

One can easily see that it is sufficient to be able to approximate x with the accuracy 2^{-k} corresponding to k binary digits. Thus, we arrive at the following definition of a computable real number (see, e.g., [8]):

Definition 1. *A real number x is called* computable *if there is an algorithm*

that, given a natural number k, generates a rational number r_k for which

$$|x - r_k| \le 2^{-k}.$$

How to store a computable number in the computer. The above definition provides a straightforward way of storing a computable real number in the actual computer: namely, once we fix the accuracy 2^{-k}, all we need to store in the corresponding rational number r_k.

What is computable: case of functions from reals to reals. In the real world, there are many dependencies between the values of different quantities. Sometimes, the corresponding dependence is *functional*, in the sense that the values $x_1, \ldots, x_n$ of some quantities x_i uniquely determine the value of some other quantity y. For example, according to the Ohm's Law $V = I \cdot R$, the voltage V is uniquely determined by the values of the current I and the resistance R.

It is reasonable to say that the corresponding function $y = f(x_1, \ldots, x_n)$ is computable if, based on the results of measuring the quantities x_i, we can predict the results of measuring y. We may not know beforehand how accurately we need to measure the quantities x_i to predict y with a given accuracy k. If the original accuracy of measuring x_i is not enough, the prediction scheme can ask for more accurate measurement results. In other words, the algorithm can ask, for each pair of natural numbers $i \le n$ and k, for a rational number r_{ik} such that $|x_i - r_{ik}| \le 2^{-k}$. The algorithm can ask for these values r_{ik} as many times as it needs, all we require is that at the end, we always get the desired prediction. Thus, we arrive at the following definition [8]:

Definition 2. *We say that a function $y = f(x_1, \ldots, x_n)$ from real numbers to real numbers is* computable *if there is an algorithm that, for all possible values x_i, given a natural number ℓ, computes a rational number s_ℓ for which $|f(x_1, \ldots, x_n) - s_\ell| \le 2^{-\ell}$. This algorithm,*

- *in addition to the usual computational steps,*
- *can also generate* requests*, i.e., pairs of natural numbers (i, k) with $i \le n$.*

As a reply to a request, the algorithm then gets a rational number r_{ik} for which $|x_i - r_{ik}| \le 2^{-k}$; this number can be used in further computations.

It is known that most usual mathematical functions are computable in this sense.

How to store a computable function in a computer. In contrast to the case of a computable real number, here, even if we know the accuracy $2^{-\ell}$ with which we need to compute the results, it is not immediately clear how we can store the corresponding function without explicitly storing the while algorithm.

To make storage easier, it is possible to take into account that in practice, for each physical quantity X_i, there are natural bounds $\underline{X}_i$ and $\overline{X}_i$: velocities are

bounded by the speed of light, distances on Earth are bounded by the Earth's size, etc. Thus, for all practical purposes, it is sufficient to only consider values $x_i \in [\underline{X}_i, \overline{X}_i]$. It turns out that for such functions, the definition of a computable function can be simplified:

Proposition 1. *For every computable function $f(x_1, \ldots, x_n)$ on a rational-valued box $[\underline{X}_1, \overline{X}_1] \times \ldots \times [\underline{X}_n, \overline{X}_n]$, there exists an algorithm that, given a natural number ℓ, computes a natural number k such that if $|x_i - x_i'| \le 2^{-k}$ for all i, then $|f(x_1, \ldots, x_n) - f(x_1', \ldots, x_n')| \le 2^{-\ell}$. This "$\ell$ to k" algorithm can be effectively constructed based on the original one.*

Comment. For reader's convenience, all the proofs are placed in the last (Proofs) section.

Because of this result, for each ℓ, to be able to compute all the values $f(x_1, \ldots, x_n)$ with the accuracy $2^{-\ell}$, it is no longer necessary to describe the whole algorithm, it is sufficient to store finitely many rational numbers. Namely:

- We use Proposition 1 to find select a value k corresponding to the accuracy $2^{-(\ell+1)}$.
- Then, for each i, we consider a finite list of rational values

$$r_i = \underline{X}_i, \quad r_i = \underline{X}_i + 2^{-k}, \quad r_i = \underline{X}_i + 2 \cdot 2^{-k}, \ldots, r_i = \overline{X}_i.$$

- For each combination of such rational values, we use the original function's algorithm to compute the value $f(r_1, \ldots, r_n)$ with accuracy $2^{-(\ell+1)}$.

These are the values we store.

Based on these stored values, we can compute all the values of the function $f(x_1, \ldots, x_n)$ with the given accuracy $2^{-\ell}$. Specifically, for each combination of computable values $(x_1, \ldots, x_n)$, we can:

- compute 2^{-k}-close rational value $r_1, \ldots, r_n$, and then
- find, in the stored list, the corresponding approximation $\widetilde{y}$ to $f(r_1, \ldots, r_n)$, i.e., the value $\widetilde{y}$ for which $|f(r_1, \ldots, r_n) - \widetilde{y}| \le 2^{-(\ell+1)}$.

Let us show that this value $\widetilde{y}$ is indeed the $2^{-\ell}$-approximation to $f(x_1, \ldots, x_n)$.

Indeed, because of our choice of ℓ, from the fact that $|x_i - r_i| \le 2^{-k}$, we conclude that $|f(x_1, \ldots, x_n) - f(r_1, \ldots, r_n)| \le 2^{-(\ell+1)}$. Thus,

$$|f(x_1, \ldots, x_n) - y| \le |f(x_1, \ldots, x_n) - f(r_1, \ldots, r_n)| + |f(r_1, \ldots, r_n) - \widetilde{y}| \le$$

$$2^{-(\ell+1)} + 2^{-(\ell+1)} = 2^{-\ell},$$

i.e., that the value $\widetilde{y}$ is indeed the desired $2^{-\ell}$-approximation to $f(x_1, \ldots, x_n)$.

A useful equivalent definition of a computable function. Proposition 1 allows us to use the following equivalent definition of a computable function.

Definition 2′. *We say that a function $y = f(x_1, \ldots, x_n)$ defined on a rational-valued box $[\underline{X}_1, \overline{X}_1] \times \ldots \times [\underline{X}_n, \overline{X}_n]$ is* computable *if there exist two algorithms:*

- *the first algorithm, given a natural number ℓ and rational values $r_1, \ldots, r_n$, computes a $2^{-\ell}$-approximation to $f(r_1, \ldots, r_n)$;*
- *the second algorithm, given a natural number ℓ, computes a natural number k such that if $|x_i - x_i'| \le 2^{-k}$ for all i, then*

$$|f(x_1, \ldots, x_n) - f(x_1', \ldots, x_n')| \le 2^{-\ell}.$$

Not all usual mathematical functions are computable. As a corollary of Definition 2′, we conclude that every computable function is continuous. Thus, discontinuous functions are not continuous, in particular, the following function:

Definition 3. *By a* step function*, we mean a function $f(x_1)$ for which:*

- *$f(x_1) = 0$ for $x < 0$ and*
- *$f(x_1) = 1$ for $x_1 \ge 0$.*

Corollary. *The step function $f(x_1)$ is not computable.*

Comment. This corollary can be proven directly, without referring to a (rather complex) proof of Proposition 2. This direct proof is also given in the Proofs section.

Consequences for representing a probability distribution: we need to go beyond computable functions. We would like to represent a general probability distribution by its cdf $F(x)$. From the purely mathematical viewpoint, this is indeed the most general representation – as opposed, e.g., to a representation that uses a probability density function which is not defined if we have a discrete variable.

Since the cdf $F(x)$ is a function, at first glance, it may make sense to say that the cdf is computable if the corresponding function $F(x)$ is computable. For many distributions, this definition makes perfect sense: the cdfs corresponding to uniform, Gaussian, and many other distributions are indeed computable functions.

However, for the degenerate random variable which is equal to $x = 0$ with probability 1, the cdf is exactly the step-function, and we have just proven that the step-function is not computable. Thus, we need to find an alternative way to represent cdfs, beyond computable functions.

What we do in this chapter. In this chapter, we provide the corresponding general description:

- first for case when we know the exact probability distribution, and
- then for the general case, when we only have a partial information about the probability distribution.

3 What We Need to Compute: A Even Briefer Reminder

The ultimate goal of all data processing is to make decision. It is known that a rational decision maker maximizes the expected value of his/her utility $u(x)$; see, e.g., [2, 4, 5, 7]. Thus, we need to be able to compute the expected values of different functions $u(x)$.

There are known procedures for eliciting from the decision maker, with any given accuracy, the utility value $u(x)$ for each x [2, 4, 5, 7]. Thus, we utility function is *computable*. We therefore need to be able to compute expected values of computable functions.

Comment. Once we are able to compute the expected values $E[u(x)]$ of different computable functions, we will thus be able to compute other statistical characteristic such as variance. Indeed, variance V can be computed as $V = E[x^2] - (E[x])^2$.

4 Simplest Case: A Single Random Variable

Description of the case. Let us start with the simplest case of a single random variable X. We would like to understand in what sense its cdf $F(x)$ is computable.

According to our general application-based approach to computability, this means that we would like to find out what we can compute about this random variable based on the observations.

What can we compute about $F(x)$? By definition, each value $F(x)$ is the *probability* that $X \le x$. So, in order to decide what we can compute about the value $F(x)$, let us recall what we can compute about probabilities in general.

What can we compute about probabilities: case of an easy-to-check event. Let us first consider the simplest situation, when we consider a probability of an easy-to-check event, i.e., an event for which, from each observation, we can tell whether this event occurred or not. Such events – like observing head when tossing a coin or getting a total of seven points when throwing two dice – are what probability textbooks start with.

In general, we cannot empirically find the exact probabilities p of such an event. Empirically, we can only estimate *frequencies* f, by observing samples of different size N. It is known that for large N, the difference $d = p - f$ between the (ideal) probability and the observed frequency is asymptotically normal, with mean $\mu = 0$ and standard deviation $\sigma = \sqrt{\frac{p \cdot (1-p)}{N}}$. We also know that for a normal distribution, situations when $|d - \mu| < 6\sigma$ are negligibly rare (with probability $< 10^{-8}$), so for all practical purposes, we can conclude that $|f - p| \le 6\sigma$.

If we believe that the probability of 10^{-8} is too high to ignore, we can take 7σ, 8σ, or $k_0 \cdot \sigma$ for an even larger value k_0. No matter what value k_0 we choose, for any given value $\delta > 0$, for sufficiently large N, we get $k_0 \cdot \sigma \le \delta$.

Thus, for each well-defined event and for each desired accuracy δ, we can find the frequency f for which $|f - p| \le \delta$. This is exactly the definition of a computable real number, so we can conclude that the probability of a well-defined event should be a computable real number.

What about the probability that $X \le x$? The desired cdf is the probability that $X \le x$. The corresponding event $X \le x$ is *not* easy to check, since we do not observe the actual value X, we only observe the measurement result $\widetilde{X}$ which is close to X.

In other words, after repeating the experiment N times, instead of N actual values $X_1, \ldots, X_n$, we only know approximate values $\widetilde{X}_1, \ldots, \widetilde{X}_n$ for which

$$|\widetilde{X}_i - X_i| \le \varepsilon$$

for some accuracy ε. Thus, instead of the "ideal" frequency $f = \mathrm{Freq}(\widetilde{X}_i \le x)$ – which is close to the desired probability $F(x) = \mathrm{Prob}(X \le x)$ – based on the observations, we get a slightly different frequency $f = \mathrm{Freq}(\widetilde{X}_i \le x)$.

What can we say about $F(x)$ based on this frequency? Since $|\widetilde{X}_i - X_i| \le \varepsilon$, the inequality $\widetilde{X}_i \le x$ implies that $X_i \le x + \varepsilon$. Similarly, if if $X_i \le x - \varepsilon$, then we can conclude that $\widetilde{X}_i \le x$. Thus, we have:

$$\mathrm{Freq}(X_i \le x - \varepsilon) \le f = \mathrm{Freq}(\widetilde{X}_i \le x) \le \mathrm{Freq}(X_i \le x + \varepsilon).$$

We have already discussed that for a sufficiently large sample, frequencies are δ-close to probabilities, so we conclude that

$$\mathrm{Prob}(X \le x - \varepsilon) - \delta \le f \le \mathrm{Prob}(\widetilde{X}_i \le x) \le \mathrm{Prob}(X_i \le x + \varepsilon) + \varepsilon.$$

So, we arrive at the following definition.

Definition 4. *We say that a cdf $F(x)$ is* computable *if there is an algorithm that, given rational values x, $\varepsilon > 0$, and $\delta > 0$, returns a rational number f for which*

$$F(x - \varepsilon) - \delta \le f \le F(x + \varepsilon) + \delta.$$

How to describe a computable cdf in a computer. How can we describe a computable cdf in a computer? The above definition kinds of prompts us to store the algorithm computing f, but algorithms may take a long time to compute. It is desirable to avoid such time-consuming computations and store only the pre-computed values – at least the pre-computed values corresponding to the given accuracy.

We cannot do this by directly following the above definition, since this definition requires us to produce an appropriate f for all infinitely many possible

rational values x. Let us show, however, that a simple and natural modification of this idea makes storing finitely many values possible.

Indeed, for two natural numbers k and ℓ, let us take $\varepsilon_0 = 2^{-k}$ and $\delta_0 = 2^{-\ell}$. On the interval $[\underline{T}, \overline{T}]$, we then select a grid $x_1 = \underline{T}$, $x_2 = \underline{T} + \varepsilon_0$, ... Due to Definition 4, for every point x_i from this grid, we can then find the value f_i for which

$$F(x_i - \varepsilon_0) - \delta_0 \le f_i \le F(x_i + \varepsilon_0) + \delta_0.$$

Let us also set up a grid 0, δ_0, $2\delta_0$, etc., on the interval $[0, 1]$ of possible values f_i, and instead of the original values f_i, let us store the closest values $\widetilde{f_i}$ from this grid.

Thus, for each pair (k, ℓ), we store a finite number of rational numbers $\widetilde{f_i}$ each of which take finite number of possible values (clearly not exceeding $1 + 1/\delta_0 = 2^\ell + 1$). Thus, for each k and ℓ, we have finitely many possible approximations of this type.

Let us show that this information is indeed sufficient to reconstruct the computable cdf, i.e., that if we have such finite-sets-of-values for all k and ℓ, then, for each rational x, $\varepsilon > 0$, and $\delta > 0$, we can algorithmically compute the value f needed in the Definition 4.

Indeed, for each ε_0 and δ_0, we can find the value x_i from the corresponding grid which is ε_0-close to x. For this x_i, we have a value $\widetilde{f_i}$ which is δ_0-close to the f_i for which

$$F(x_i - \varepsilon_0) - \delta_0 \le f_i \le F(x_i + \varepsilon_0) + \delta_0.$$

Thus, we have

$$F(x_i - \varepsilon_0) - 2\delta_0 \le \widetilde{f_i} \le F(x_i + \varepsilon_0) + 2\delta_0.$$

From $|x_i - x| \le \varepsilon_0$, we conclude that $x_i + \varepsilon_0 \le x + 2\varepsilon_0$ and $x - 2\varepsilon_0 \le x_i - \varepsilon_0$ and thus, that $F(x - 2\varepsilon_0) \le F(x_i - \varepsilon_0)$ and $F(x_i + \varepsilon_0) \le F(x + 2\varepsilon_0)$. Hence,

$$F(x - 2\varepsilon_0) - 2\delta_0 \le \widetilde{f_i} \le F(x + 2\varepsilon_0) + 2\delta_0.$$

So, if we take ε_0 and δ_0 for which $2\varepsilon_0 \le \varepsilon$ and $2\delta_0 \le \delta$, then we get

$$F(x - \varepsilon) \le F(x - 2\varepsilon_0) - 2\delta_0 \le \widetilde{f_i} \le F(x + 2\varepsilon_0) + 2\delta_0 \le F(x + \varepsilon) + \delta,$$

i.e., we have the desired double inequality

$$F(x - \varepsilon) - \delta \le \widetilde{f_i} \le F(x + \varepsilon) + \delta,$$

with $f = \widetilde{f_i}$.

Equivalent definitions. Anyone who seriously studied mathematical papers and books have probably noticed that, in addition to definitions of different notions and theorems describing properties of these notions, these papers and books often have, for many of these notions, several different but mathematically

equivalent definitions. The motivation for having several definitions is easy to understand: if we have several equivalent definitions, then in each case, instead of trying to use the original definition, we can select the one which is the most convenient to use. In view of this, let us formulate several equivalent definitions of a computable cdf.

Definition 4′. *We say that a cdf $F(x)$ is* computable *if there is an algorithm that, given rational values x, $\varepsilon > 0$, and $\delta > 0$, returns a rational number f which is δ-close to $F(x')$ for some x' for which $|x' - x| \le \varepsilon$.*

Proposition 2. *Definitions 4 and 4′ are equivalent to each other.*

To get the second equivalent definition, we start with the pairs $(x_i, \widetilde{f_i})$ that we decided to use to store the computable cdf. When $f_{i+1} - f_i > \delta$, we add intermediate pairs

$$(x_i, f_i + \delta), (x_i, f_i + 2\delta), \ldots, (x_i, f_{i+1}).$$

We can say that the resulting finite set of pairs is (ε, δ)-*close* to the graph $\{(x, y) : F(x - 0) \le y \le F(x)\}$ in the following sense.

Definition 5. *Let $\varepsilon > 0$ and $\delta > 0$ be two rational numbers.*

- *We say that pairs (x, y) and (x', y') are* (ε, δ)-close *if $|x - x'| \le \varepsilon$ and $|y - y'| \le \delta$.*
- *We say that the sets S and S' are* (ε, δ)-close *if:*
  - *for every $s \in S$, there is a (ε, δ)-close point $s' \in S'$;*
  - *for every $s' \in S'$, there is a (ε, δ)-close point $s \in S$.*

Comment. This definition is similar to the definition of ε-closeness in *Hausdorff metric*, where the two sets S and S' are ε-close if:

- for every $s \in S$, there is a ε-close point $s' \in S'$;
- for every $s' \in S'$, there is a ε-close point $s \in S$.

Definition 4″. *We say that a cdf $F(x)$ is* computable *if there is an algorithm that, given rational values $\varepsilon > 0$ and $\delta > 0$, produces a finite list of pairs which is* (ε, δ)-close *to the graph $\{(x, y) : F(x - 0) \le y \le F(x)\}$.*

Proposition 3. *Definition 4″ is equivalent to Definitions 4 and 4′.*

Comment. Proof of Proposition 3 is similar to the above argument that our computer representation is sufficient for describing a computable cdf.

What can be computed: a positive result for the 1-D case. We are interested in computing the expected value $E_{F(x)}[u(x)]$ for computable functions $u(x)$. For this problem, we have the following result:

Theorem 1. *There is an algorithm that:*

- *given a computable cdf $F(x)$,*
- *given a computable function $u(x)$, and*
- *given (rational) accuracy $\delta > 0$,*

computes $E_{F(x)}[u(x)]$ with accuracy δ.

5 What If We Only Have Partial Information about the Probability Distribution?

Need to consider mixtures of probability distributions. The above result deals with the case when we have a single probability distribution, and by observing larger and larger samples we can get a better and better understanding of the corresponding probabilities. This corresponds to the ideal situation when all sub-samples have the same statistical characteristics. In practice, this is rarely the case. What we often observe is, in effect, a mixture of several samples with slightly different probabilities. For example, if we observe measurement errors, we need to take into account that a minor change in manufacturing a measuring instrument can cause a slight different in the resulting probability distribution of measurement errors.

In such situations, instead of a *single* probability distribution, we need to consider a *set* of possible probability distributions.

Another case when we need to consider a set of distributions is when we only have partial knowledge about the probabilities. In all such cases, we need to process sets of probability distributions. To come up with an idea of how to process such sets, let us first recall how sets are dealt with in computations. For that, we will start with the simplest case: sets of numbers (or tuples).

Computational approach to sets of numbers: reminder. In the previous sections, we considered computable numbers and computable tuples (and computable functions). A number (or a tuple) corresponds to the case when we have a complete information about the value of the corresponding quantity (quantities). In practice, we often only have *partial* information about the actual value. In this case, instead of *single* value, we have a *set* of possible values. How can we represent such sets in a computer?

At first glance, this problem is complex, since there are usually infinitely many possible numbers – e.g., all numbers from an interval, and it is not clear how to represent infinitely many number in a computer – which is only capable of storing finite number of bits.

However, a more detailed analysis shows that the situation is not that hopeless: infinite number of values only appears in the idealized case when we assume that all the measurements are absolutely accurate and thus, produce the exact value. In practice, as we have mentioned, measurements have uncertainty and

thus, with each measuring instrument, we can only distinguish between finitely many possible outcomes.

So, for each set S of possible values, for each accuracy ε, we can represent this set by a finite list S_ε of possible ε-accurate measurement results. This finite list has the following two properties:

- each value $s_i \in S_\varepsilon$ is the result of an ε-accurate measurement and is, thus, ε-close to some value $s \in S$;
- vice versa, each possible value $s \in S$ is represented by one of the possible measurement results, i.e., for each $s \in S$, there exists an ε-close value

$$s_i \in S_\varepsilon.$$

Comment. An attentive reader may recognize that these two conditions have already been mentioned earlier – they correspond to ε-closeness of the sets S and S_ε in terms of Hausdorff metric.

Thus, we naturally arrive at the following definition.

Definition 6. *A set S is called* computable *if there is an algorithm that, given a rational number $\varepsilon > 0$, generates a finite list S_ε for which:*

- *each element $s \in S$ is ε-close to some element from this list, and*
- *each element from this list is ε-close to some element from the set S.*

Comment. In mathematics, sets which can be approximated by finite sets are known as *compact sets*. Because of this, computable sets are also known as *computable compacts*; see, e.g., [1].

So how do we describe partial information about the probability distribution. We have mentioned that for each accuracy (ε, δ), all possible probability distributions can be represented by the corresponding finite lists – e.g., if we use Definition $4''$, as lists which are (ε, δ)-close to the corresponding cdf $F(x)$.

It is therefore reasonable to represent a set of probability distributions – corresponding to partial knowledge about probabilities – by finite lists of such distributions.

Definition 7. *A set $\mathbf{S}$ of probability distributions is called* computable *if there is an algorithm that, given rational numbers $\varepsilon > 0$ and $\delta > 0$, generates a finite list $\mathbf{S}_{\varepsilon,\delta}$ of computable cdfs for which:*

- *each element $s \in \mathbf{S}$ is (ε, δ)-close to some element from this list, and*
- *each element from this list is (ε, δ)-close to some element from the set $\mathbf{S}$.*

What can be computed? For the same utility function $u(x)$, different possible probability distributions lead, in general, to different expected values. In

such a situation, it is desirable to find the *range* $E_{\mathbf{S}}[u(x)] = [\underline{E}_{\mathbf{S}}[u(x), \overline{E}_S[u(x)]]$ of possible values of $E_{F(x)}[u(x)]$ corresponding to all possible probability distributions $F(x) \in \mathbf{S}$:

$$\underline{E}_{\mathbf{S}}[u(x)] = \min_{F(x)\in\mathbf{S}} E_{F(x)}[u(x)]; \quad \overline{E}_{\mathbf{S}}[u(x)] = \max_{F(x)\in\mathbf{S}} E_{F(x)}[u(x)].$$

It turns out that, in general, this range is also computable:

Theorem 2. *There is an algorithm that:*

- *given a computable set* $\mathbf{S}$ *of probability distributions,*
- *given a computable function* $u(x)$*, and*
- *given (rational) accuracy* $\delta > 0$*,*

computes the endpoints of the range $E_{\mathbf{S}}[u(x)]$ *with accuracy* δ*.*

Comment. This result follows from Theorem 1 and from the known fact that there is a general algorithm for computing maximum and minimum of a computable function on a computable compact; see, e.g., [1].

6 What to Do in a General Case (Not Necessarily 1-D)

Need to consider a general case. What if we have a joint distribution of several variable? A random process – i.e., a distribution on the set of functions of one variable? A random field – a probability distribution on the set of functions of several variables? A random operator? A random set?

In all these cases, we have a natural notion of a distance (metric) which is computable, so we have probability distribution on a computable metric space M.

Situations when we know the exact probability distribution: main idea. In the general case, the underlying metric space M is not always ordered, so we cannot use cdf $F(x) = \text{Prob}(X \le x)$ to describe the corresponding probability distribution.

However, what we observe and measure are still numbers – namely, each measurement can be described by a computable function $g : M \to \mathbb{R}$ that maps each state $m \in M$ into a real number. By performing such measurements many times, we can get the frequencies of different values of $g(x)$. Thus, we arrive at the following definition.

Definition 8. *We say that a probability distribution on a computable metric space is* computable *if there exists an algorithm, that, given:*

- *a computable real-valued function $g(x)$ on M, and*
- *rational numbers y, $\varepsilon > 0$, and $\delta > 0$,*

returns a rational number f which is ε-close to the probability $\mathrm{Prob}(g(x) \le y')$ *for some y' which is δ-close to y.*

How can we represent this information in a computer? Since M is a computable set, for every ε, there exists an *ε-net* $x_1, \ldots, x_n$ for M, i.e., a finite list of points for which, for every $x \in M$, there exists an ε-close point x_i from this list, thus

$$X = \bigcup_i B_\varepsilon(x_i), \text{ where } B_\varepsilon(x) \stackrel{\text{def}}{=} \{x' : d(x, x') \le \varepsilon\}.$$

For each computable element x_0, by applying the algorithm from Definition 8 to a function $g(x) = d(x, x_0)$, we can compute, for each ε_0 and δ_0, a value f which is close to $\mathrm{Prob}(B_{\varepsilon'}(x_0))$ for some ε' which is δ_0-close to ε_0.

In particular, by taking $\delta_0 = 2^{-k}$ and $\varepsilon_0 = \varepsilon + 2 \cdot 2^{-k}$, we can find a value f' which is 2^{-k}-close to $\mathrm{Prob}(B_{\varepsilon'}(x_0))$ for some $\varepsilon' \in [\varepsilon + 2^{-k}, \varepsilon + 3 \cdot 2^{-k}]$. Similarly, by taking $\varepsilon_0' = \varepsilon + 5 \cdot 2^{-k}$, we can find a value f'' which is 2^{-k}-close to $\mathrm{Prob}(B_{\varepsilon''}(x_0))$ for some $\varepsilon'' \in [\varepsilon + 4 \cdot 2^{-k}, \varepsilon + 6 \cdot 2^{-k}]$.

We know that when we have $\varepsilon < \varepsilon' < \varepsilon''$ and $\varepsilon'' \to \varepsilon$, then

$$\mathrm{Prob}(B_{\varepsilon''}(x_0) - B_{\varepsilon'}(x_0)) \to 0,$$

so the values f' and f'' will eventually become close. Thus, by taking $k = 1, 2, \ldots$, we will eventually compute the number f_1 which is close to $\mathrm{Prob}(B_{\varepsilon'}(x_1))$ for all ε' from some interval $[\underline{\varepsilon}_1, \overline{\varepsilon}_1]$ which is close to ε (and for which $\underline{\varepsilon} > \varepsilon$).

We then:

- select f_2 which is close to $\mathrm{Prob}(B_{\varepsilon'}(x_1) \cup B_{\varepsilon'}(x_2))$ for all ε' from some interval $[\underline{\varepsilon}_2, \overline{\varepsilon}_2] \subseteq [\underline{\varepsilon}_1, \overline{\varepsilon}_1]$,
-
- select f_3 which is close to $\mathrm{Prob}(B_{\varepsilon'}(x_1) \cup B_{\varepsilon'}(x_2) \cup B_{\varepsilon'}(x_3))$ for all ε' from some interval $[\underline{\varepsilon}_3, \overline{\varepsilon}_3] \subseteq [\underline{\varepsilon}_2, \overline{\varepsilon}_2]$,
- etc.

At the end, we get approximations $f_i - f_{i-1}$ to probabilities of the sets

$$S_i \stackrel{\text{def}}{=} B_\varepsilon(x_i) - (B_\varepsilon(x_1) \cup \ldots \cup B_\varepsilon(x_{i-1}))$$

for all ε from the last interval $[\underline{\varepsilon}_n, \overline{\varepsilon}_n]$.

These approximations $f_i - f_{i-1}$ form the information that we store about the probability distribution – as well as the values x_i.

What can we compute? It turns out that we can compute the expected value $E[u(x)]$ of any computable function:

Theorem 3. *There is an algorithm that:*

- *given a computable probability distribution on a computable metric space,*
- *given a computable function $u(x)$, and*
- *given (rational) accuracy $\delta > 0$,*

computes the expected value $E[u(x)]$ with accuracy δ.

What if we have a set of possible probability distributions? In the case of partial information about the probabilities, we have a set $\mathbf{S}$ of possible probability distributions.

In the computer, for any given accuracies ε ad δ, each computable probability distribution is represented by the values $f_1, \ldots, f_n$. A computable set of distributions can be then defined by assuming that, for every ε and δ, instead of a single tuples $(f_1, \ldots, f_n)$, we have a *computable set* of such tuples.

In this case, similar to the 1-D situation, it is desirable to find the *range* $E_{\mathbf{S}}[u(x)] = [\underline{E}_{\mathbf{S}}[u(x)], \overline{E}_S[u(x)]]$ of possible values of $E_P[u(x)]$ corresponding to all possible probability distributions $P \in \mathbf{S}$:

$$\underline{E}_{\mathbf{S}}[u(x)] = \min_{P \in \mathbf{S}} E_{F(x)}[u(x)]; \quad \overline{E}_{\mathbf{S}}[u(x)] = \max_{P \in \mathbf{S}} E_{F(x)}[u(x)].$$

In general, this range is also computable:

Theorem 4. *There is an algorithm that:*

- *given a computable set $\mathbf{S}$ of probability distributions,*
- *given a computable function $u(x)$, and*
- *given (rational) accuracy $\delta > 0$,*

computes the endpoints of the range $E_{\mathbf{S}}[u(x)]$ with accuracy δ.

Comment. Similarly to Theorem 2, this result follows from Theorem 3 and from the known fact that there is a general algorithm for computing maximum and minimum of a computable function on a computable compact [1].

7 Proofs

Proof of Proposition 1.

1°. Once we can approximate a real number x with an arbitrary accuracy, we can always find, for each k, a 2^{-k}-approximation r_k of the type $\dfrac{n_k}{2^k}$ for some integer n_k.

Indeed, we can first find a rational number r_{k+1} for which $|x - r_{k+1}| \le 2^{-(k+1)}$, and then take $r_k = \dfrac{n_k}{2^k}$ where n_k is the integer which is the closest to the rational number $2^k \cdot r_{k+1}$. Indeed, for this closest integer, we have

$$|2^k \cdot r_{k+1} - n_k| \le 0.5.$$

By dividing both sides of this inequality by 2^k, we get $|r_{k+1} - r_k| = \left|r_{k+1} - \dfrac{n_k}{2^k}\right| \le 2^{-(k+1)}$, and thus, indeed,

$$|x - r_k| \le |x - r_{k+1}| + |r_{k+1} - r_k| \le 2^{-(k+1)} + 2^{-(k+1)} = 2^{-k}.$$

2°. Because of Part 1 of this proof, it is sufficient to consider situations in which, as a reply to all its requests (i, k), the algorithm receives the approximate value r_{ik} of the type $\dfrac{n_{ik}}{2^k}$.

3°. Let us prove, by contradiction, that for given ℓ, there exists a value $k_{\max}$ that bounds, from above, the indices k in the all the requests (i, k) that this algorithm makes when computing a $2^{-\ell}$-approximation to $f(x_1, \ldots, x_n)$ on all possible inputs.

If this statement is not true, this means that for every natural number x, there exist a tuple $x^{(k)} = (x_1^{(k)}, \ldots, x_n^{(k)})$ for which this algorithm requests an approximation of accuracy at least 2^{-k} to at least one of the values $x_i^{(k)}$.

Overall, we have infinitely many tuples corresponding to infinitely many natural numbers. As a reply to each request (i, k), we get a rational number of the type $r_{ik} = \dfrac{n_{ik}}{2^k}$. For each natural number m, let us consider the value $\dfrac{p_i}{2^m}$ which is the closest to r_{ik}. There are finitely many possible tuples $(p_1, \ldots, p_n)$, so at least one of these tuples occurs infinitely many times.

Let us select such a tuple t_1 corresponding to $m = 1$. Out of infinitely many cases when we get an approximation to this tuple, we can select, on the level $m = 2$, a tuple t_2 for which we infinitely many times request the values which are 2^{-2}-close to this tuple, etc. As a result, we get a sequence of tuples t_m for which $|t_m - t_{m+1}| \le 2^{-m} + 2^{-(m+1)}$.

This sequence of tuples converges. Let us denote its limit by $t = (t_1, \ldots, t_n)$. For this limit, for each k, the algorithm follows the same computation as the k-th tuple and thus, will request some value with accuracy $\le 2^{-k}$. Since this is true for every k, this means that this algorithm will never stop – and we assumed that our algorithm always stops. This contradiction proves that there indeed exists an upper bound $k_{\max}$.

4°. How can we actually find this $k_{\max}$? For that, let us try values $m = 1, 2, \ldots$ For each m, we apply the algorithm $f(r_1, \ldots, r_n)$ to all possible combinations of values of the type $r_i \dfrac{p_i}{2^m}$; in the original box, for each m, there are finitely many such tuples. For each request (i, k), we return the number of the type $\dfrac{n_{ik}}{2^k}$ which

is the closest to t_i. When we reach the value $m = k_{\max}$, then, by definition of $k_{\max}$, this would mean that our algorithm never requires approximations which are more accurate than 2^{-m}-accurate ones.

In this case, we can then be sure that we have reached the desired value $k_{\max}$: indeed, for all possible tuples $(x_1, \ldots, x_n)$, this algorithm will never request values beyond this m-th approximation – and we have shown it for all possible combinations of such approximations. The proposition is proven.

Direct proof of the Corollary to Proposition 1. The non-computability of the step function can be easily proven by contradiction. Indeed, suppose that there exists an algorithm that computes this function. Then, for $x_1 = 0$ and $\ell = 2$, this algorithm produces a rational number s_ℓ which is 2^{-2}-close to the value $f(0) = 1$ and for which, thus, $s_\ell \geq 0.75$. This algorithm should work no matter which approximate values r_{1k} it gets – as long as these values are 2^{-k}-close to x_1. For simplicity, let us consider the case when all these approximate values are 0s: $r_{1k} = 0$.

This algorithm finishes computations in finitely many steps, during which it can only ask for the values of finitely many such approximations; let us denote the corresponding accuracies by $k_1, \ldots, k_m$, and let $K = \max(k_1, \ldots, k_m)$ be the largest of these natural numbers. In this case, all the information that this algorithm uses about the actual value x is that this value satisfies all the corresponding inequalities $|x_1 - r_{1k_j}| \leq 2^{-k_j}$, i.e., $|x_1| \leq 2^{-k_j}$. Thus, for any other value x_1' that satisfies all these inequalities, this algorithm returns the exact same value $s_\ell \geq 0.75$. In particular, this will be true for the value $x_1' = -2^{-K}$. However, for this negative value x_1', we should get $f(x_1') = 0$, and thus, the desired inequality $|f(x_1') - y_\ell| \leq 2^{-2}$ is no longer satisfied. This contradiction proves that the step function is not computable.

Proof of Proposition 2. It is easy to show that Definition 4′ implies Definition 4. Indeed, if f is δ-close to $F(x')$ for some $x' \in [x - \varepsilon, x + \varepsilon]$, i.e., if $F(x') - \delta \leq f \leq F(x') + \delta$, then, due to $x - \varepsilon \leq x' \leq x + \varepsilon$, we get $F(x - \varepsilon) \leq F(x')$ and $F(x') \leq F(x + \varepsilon)$ and thus, that

$$F(x - \varepsilon) \leq F(x') - \delta \leq f \leq F(x') + \delta \leq F(x + \varepsilon) + \delta,$$

i.e., the desired inequality

$$F(x - \varepsilon) \leq f \leq F(x + \varepsilon) + \delta.$$

Vice versa, let us show that Definition 4 implies Definition 4′. Indeed, we know that $F(x + \varepsilon) - F(x + \varepsilon/3) \to 0$ as $\varepsilon \to 0$. Indeed, this difference is the probability of X being in the set $\{X : x + \varepsilon/3 \leq X \leq x + \varepsilon\}$, which is a subset of the set $S_\varepsilon \stackrel{\text{def}}{=} \{X : x < X \leq x + \varepsilon\}$. The sets S_ε form a nested family with an empty intersection, thus their probabilities tend to 0 and thus, the probabilities of their subsets also tend to 0.

Due to Proposition 4, for each $k = 1, 2, \ldots$, we can take $\varepsilon_k = \varepsilon \cdot 2^{-k}$ and find f_k and f_k' for which

$$F(x + \varepsilon_k/3) - \delta/4 \leq f_k \leq F(x + (2/3) \cdot \varepsilon_k) + \delta/4$$

and

$$F(x + (2/3) \cdot \varepsilon_k) - \delta/4 \le f'_k \le F(x + \varepsilon_k) + \delta/4.$$

From these inequalities, we conclude that

$$-\delta/2 \le f'_k - f_k \le F(x + \varepsilon_k) - F(x + \varepsilon_k/3) + \delta/2.$$

Since $F(x + \varepsilon_k) - F(x + \varepsilon_k/3) \to 0$ as $k \to \infty$, for sufficiently large k, we will have $F(x+\varepsilon_k)-F(x+\varepsilon_k/3) \le \delta/4$ and thus, $|f'_k - f_k| \le (3/4) \cdot \delta$. By computing the values f_k and f'_k for $k = 1, 2, \ldots$, we will eventually reach an index k for which this inequality is true. Let us show that this f_k is then δ-close to $F(x')$ for $x' = x + (2/3) \cdot \varepsilon_k$ (which is ε_k-close – and thus, ε-close – to x).

Indeed, we have

$$f_k \le F(x + (2/3) \cdot \varepsilon_k) + \delta/4 \le F(x + (2/3) \cdot \varepsilon_k) + \delta.$$

On the other hand, we have

$$F(x + (2/3) \cdot \varepsilon_k) - \delta/4 \le f'_k \le f_k + (3/4) \cdot \delta$$

and thus,

$$F(x + (2/3) \cdot \varepsilon_k) - \delta \le f_k \le F(x + (2/3) \cdot \varepsilon_k) + \delta.$$

The equivalence is proven.

Proof of Theorem 1. We have shown, in Proposition 1, that every computable function $u(x)$ is computably continuous, in the sense that for every $\delta_0 > 0$, we can compute $\varepsilon > 0$ for which $|x - x'| \le \varepsilon$ implies $|u(x) - u(x')| \le \delta_0$.

In particular, if we take ε corresponding to $\delta_0 = 1$, and take the ε-grid $x_1, \ldots, x_i, \ldots$, then we conclude that each value $u(x)$ is 1-close to one of the values $u(x_i)$ on this grid. So, if we compute the 1-approximations $\widetilde{u}_i$ to the values $u(x_i)$, then each value $u(x)$ is 2-close to one of these values $\widetilde{u}_i$. Thus, $\max\limits_x |u(x)| \le U \stackrel{\text{def}}{=} \max\limits_i \widetilde{u}_i + 2$. So, we have a computable bound $U \ge 2$ for the (absolute value) of the computable function $u(x)$.

Let us once again use computable continuity. This time, we select ε corresponding to $\delta_0 = \delta/4$, and take an x-grid $x_1, \ldots, x_i, \ldots$ with step $\varepsilon/4$. Let G be the number of points in this grid.

According to the equivalent form (Definition $4'$) of the definition of computable cdf, for each of these grid points x_i, we can compute the value f_i which is $(\delta/(4U \cdot G))$-close to $F(x'_i)$ for some x'_i which is $(\varepsilon/4)$-close to x_i.

The function $u(x)$ is $(\delta/4)$-close to a piece-wise constant function $u'(x)$ which is equal to $u(x_i)$ for $x \in (x'_i, x'_{i+1}]$. Thus, their expected values are also $(\delta/4)$-close: $|E[u(x)] - E[u'(x)]| \le \delta/4$.

Here, $E[u'(x)] = \sum\limits_i u(x_i) \cdot (F(x'_{i+1}) - F(x'_i))$. But $F(x'_i)$ is $(\delta/(4U \cdot G))$-close to f_i and $F(x'_{i+1})$ is $(\delta/(4U \cdot G))$-close to f_{i+1}. Thus, each difference $F(x'_{i+1}) - F(x'_i)$ is $(\delta/(2U \cdot G))$-close to the difference $f_{i+1} - f_i$.

Since $|u(x_i)| \le U$, we conclude that each term $u(x_i) \cdot (F(x'_{i+1}) - F(x'_i))$ is $(\delta/(2G))$-close to the computable term $u(x_i) \cdot (f_{i+1} - f_i)$. Thus, the sum of G

such terms – which is equal to $E[u'(x)]$ – is $(\delta/2)$-close to the computable sum $\sum_i u(x_i)\cdot(f_{i+1}-f_i)$. Since $E[u'(x)]$ is, in its turn, $(\delta/4)$-close to desired expected value $E[u(x)]$, we thus conclude that the above computable sum

$$\sum_i u(x_i)\cdot(f_{i+1}-f_i)$$

is indeed a δ-approximation to the desired expected value.

The theorem is proven.

Proof of Theorem 3. The proof is similar to the proof of Theorem 1: we approximate the function $u(x)$ by a $(\delta/2)$-close function $u'(x)$ which is piecewise constant, namely, which is equal to a constant $u_i = u(x_i)$ on each set

$$S_i = B_\varepsilon(x_i) - (B_\varepsilon(x_1) \cup \ldots \cup B_\varepsilon(x_{i-1})).$$

The expected value of the function $u'(x)$ is equal to $E[u'(x)] = \sum_i u_i \cdot \mathrm{Prob}(S_i)$.

The probabilities $\mathrm{Prob}(S_i)$ can be computed with any given accuracy, in particular, with accuracy $\delta/(2U\cdot n)$, thus enabling us to compute $E[u'(x)]$ with accuracy $\delta/2$.

Since the functions $u(x)$ and $u'(x)$ are $(\delta/2)$-close, their expected values are also $(\delta/2)$-close. So, a $(\delta/2)$-approximation to $E[u'(x)]$ is the desired δ-approximation to $E[u(x)]$.

The theorem is proven.

Acknowledgments

This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721. The authors are thankful to Walid Taha and to all the participants of the Second Hybrid Modeling Languages Meeting HyMC (Houston, Texas, May 7–8, 2015) for valuable discussions.

References

[1] E. Bishop and D. Bridges, *Constructive Analysis*, Springer Verlag, Heidelberg, 1985.

[2] P. C. Fishburn, *Utility Theory for Decision Making*, John Wiley & Sons Inc., New York, 1969.

[3] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and Feasibility of Data Processing and Interval Computations*, Kluwer, Dordrecht, 1997.

[4] R. D. Luce and R. Raiffa, *Games and Decisions: Introduction and Critical Survey*, Dover, New York, 1989.

[5] H. T. Nguyen, O. Kosheleva, and V. Kreinovich, "Decision making beyond Arrow's 'impossibility theorem', with the analysis of effects of collusion and mutual attraction", *International Journal of Intelligent Systems*, 2009, Vol. 24, No. 1, pp. 27–47.

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

[7] H. Raiffa, *Decision Analysis*, Addison-Wesley, Reading, Massachusetts, 1970.

[8] K. Weihrauch, *Computable Analysis*, Springer Verlag, Berlin, 2000.