5-2003

Fast Quantum Algorithms for Handling Probabilistic, Interval, and Fuzzy Uncertainty

Mark Martinez

Luc Longpre
The University of Texas at El Paso, longpre@utep.edu

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

Scott A. Starks
The University of Texas at El Paso, sstarks@utep.edu

Hung T. Nguyen

Comments:
Technical Report: UTEP-CS-03-16
Published in: *Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society NAFIPS'03*, Chicago, Illinois, July 24-26, 2003, pp. 395-400.

Fast Quantum Algorithms for Handling Probabilistic, Interval, and Fuzzy Uncertainty

Mark Martinez, Luc Longpré
Vladik Kreinovich, Scott A. Starks
NASA Pan-American Center for Earth and Environmental Studies (PACES)
University of Texas at El Paso
El Paso, TX 79968, USA
vladik@cs.utep.edu

Hung T. Nguyen
Department of Mathematical Sciences
New Mexico State University
Las Cruces, NM 88003, USA
hunguyen@nmsu.edu

Abstract

We show how quantum computing can speed up computations related to processing probabilistic, interval, and fuzzy uncertainty.

1. Introduction

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., [5, 6, 17]) 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 fuzzy 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, and fuzzy – 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.

2 What Is Known: Quantum Algorithms That We Can Use

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., [1, 17]. 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., [2, 17]. 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 [7]; for further developments, see, e.g., [8, 15, 18].

The traditional Monte-Carlo method for computing the mean consists of picking M random values and averaging them. It is a well known fact [19, 21], 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 $f(x_1, \ldots, x_n)$.

3 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., [19, 21]). 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

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

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., [20].

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.

4 Quantum Algorithms for Interval Computations

Problem. In interval computations (see, e.g., [9, 10, 14]), 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

$$[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i], \qquad (4)$$

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 $|y - \widetilde{y}|$.

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

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

It is known [20] 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.

General case. Known results about the computational complexity of interval computations (see, e.g., [12]) 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.

5 Quantum Algorithms for Fuzzy Computations

Formulation of the problem. In fuzzy data processing (see, e.g., [11, 16]), the main objective is: given:

- fuzzy numbers $X_1, \ldots, X_n$ characterizing our uncertainty about 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 fuzzy number $Y = f(X_1, \ldots, X_n)$ that describes the resulting knowledge about y.

First quantum algorithm: idea. One possibility of using quantum computing to speed up fuzzy data processing comes from the known fact that a fuzzy number can be represented as a nested family of intervals (α-cuts corresponding to different values of α). Therefore, in principle, we can perform fuzzy data processing by performing interval computations on the corresponding α-cuts.

Second quantum algorithm: idea. It is also possible to use quantum computing more directly. Indeed, the formula for computing the membership function of $Y = f(X_1, \ldots, X_n)$ – based on extension principle – requires that we take min over all possible combinations of $x_1, \ldots, x_n$. This computation is the most time-consuming part of fuzzy data processing; we can decrease the running time to square root of it if we use the quantum algorithm for computing minimum.

6 Quantum Algorithms for the Case When We Have Several Different Types of Uncertainty

Problem. 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., [19]. 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.

What is known. 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 [3, 4], 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.

Monte-Carlo speed-up for $\underline{V}$. 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 [3, 3] 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

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

  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

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

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

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

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 $\le 2n$ values V_k'; this requires $O(n)$ steps.

Quantum speed-up for $\underline{V}$. 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.

Speed-up for $\overline{V}$. 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 $x_{(1)} \le x_{(2)} \le \ldots \le x_{(2n)}$. 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 $\le 2^C \cdot 2n = O(n)$ 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.

7 Can Quantum Computers Be Still Useful When There Are Not Yet Enough Qubits?

Formulation of the problem. 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 [13]. 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.

Grover's algorithm: result. 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.

Acknowledgments.

This work was supported in part by NASA grant NCC5-209, by the AFOSR grant F49620-00-1-0365, by NSF grants EAR-0112968 and EAR-0225670, by the ARL grant DATM-05-02-C-0046, by a research grant from Sandia National Laboratories as part of the Department of Energy Accelerated Strategic Computing Initiative (ASCI), and by the IEEE/ACM SC2001 and SC2002 Minority Serving Institutions Participation Grants.

One of the authors (V.L.) is thankful to Bart Kosko for valuable discussions.

References

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

[2] C. Dürr and P. Hoyer, *A quantum algorithm for finding the minimum*, 1996; LANL arXiv:quant-ph/9607014

[3] S. Ferson, L. Ginzburg, V. Kreinovich, L. Longpré, and M. Aviles, Computing Variance for Interval Data is NP-Hard, *ACM SIGACT News* 33(2):108–118, 2002.

[4] S. Ferson, L. Ginzburg, V. Kreinovich, and M. Aviles, Exact Bounds on Sample Variance of Interval Data, *Abstracts of 2002 SIAM Workshop on Validated Computing*, Toronto, Canada, May 23–25, 2002, 67–69.

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

[6] L. K. Grover, Quantum mechanics helps in searching for a needle in a haystack, *Phys. Rev. Lett.*, 79(2):325–328, 1997.

[7] L. Grover, A framework for fast quantum mechanical algorithms, *Phys. Rev. Lett.* 80:4329–4332, 1998.

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

[9] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer-Verlag, London, 2001.

[10] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer, Dordrecht, 1996.

[11] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, Upper Saddle River, NJ, 1995.

[12] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1997.

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

[14] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

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

[16] H. T. Nguyen and E. A. Walker, *First Course in Fuzzy Logic*, CRC Press, Boca Raton, FL, 1999.

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

[18] E. Novak, Quantum Complexity of integration, *J. Complexity*, 17:2–16, 2001.

[19] S. Rabinovich, *Measurement Errors: Theory and Practice*, American Institute of Physics, N.Y., 1993.

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

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