Quantum Computation Techniques for Gauging Reliability of Interval and Fuzzy Data

Luc Longpré and Christian Servin
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968
Contact email christians@miners.utep.edu

***Abstract*—In traditional interval computations, we assume that the interval data corresponds to guaranteed interval bounds, and that fuzzy estimates provided by experts are correct. In practice, measuring instruments are not 100% reliable, and experts are not 100% reliable, we may have estimates which are "way off", intervals which do not contain the actual values at all. Usually, we know the percentage of such outlier un-reliable measurements. However, it is desirable to check that the reliability of the actual data is indeed within the given percentage. The problem of checking (gauging) this reliability is, in general, NP-hard; in reasonable cases, there exist feasible algorithms for solving this problem. In this paper, we show that quantum computations techniques can drastically speed up the computation of reliability of given data.**

I. Two main sources of information about the real-world objects

In many practical situations, we want to know the state of the real-world objects and/or systems. In science, we are simply interested in this state. For example, we may want to know the level of water in the river, so that we will be able to predict the possible floods. In engineering, we need the information about the state of the world to decide on the best way to favorably change the situation: e.g., how to build a dam to prevent flooding.

To describe the state of the objects and/or systems, we must describe the values of the physical quantities that characterize this state. To get the most accurate and the most reliable estimate of each quantity, we can measure it – directly or indirectly. In many cases, it is too difficult or too expensive to measure all the quantities; in such situations, we can ask the experts to estimate the values of these quantities. Measurements and expert estimates are thus the two main sources of information about the real-world objects and systems.

II. Measurement uncertainty and interval data

Measurements are usually more accurate than expert estimates, but they are never 100% accurate. The result $\widetilde{x}$ of a measurement is usually somewhat different from the actual (unknown) value x of the quantity of interest.

Usually, the manufacturer of the measuring instrument provides us with an upper bound Δ on the absolute value of the measurement error $\Delta x \stackrel{\text{def}}{=} \widetilde{x} - x$: $|\Delta x| \le \Delta$. Because of this bound, once we know the measurement result $\widetilde{x}$, we can conclude that the actual (unknown) value x belongs to the interval $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$.

In some situations, we also know the probabilities of different values $\Delta x \in [-\Delta, \Delta]$. In this case, we can use the standard statistical techniques used in science and engineering to process the corresponding uncertainty; see, e.g., [16]. However, in many practical situations, we do not know these probabilities, we only know the upper bound Δ. In these situations, the only information that we have about x is that x belongs to the interval $\mathbf{x} \stackrel{\text{def}}{=} [\widetilde{x} - \Delta, \widetilde{x} + \Delta]$. In such situations, we need to process this interval data; see, e.g., [6].

III. Expert estimates and fuzzy data

When measurement is not possible, we can use experts to estimate the values of the desired quantity. Expert estimates are never exact, they are approximate estimates $\widetilde{x}$ of the desired quantity x: $\Delta x = \widetilde{x} - x \neq 0$. Of course, in contrast to the measuring instruments for which the manufacturer provides us with an upper bound on the measurement error, there is no guarantee of expert's accuracy. Instead of the exact 100% bounds on $|\Delta x|$, we can provide bounds which are valid with some degree of certainty. This degree of certainty is usually described by a number from the interval $[0, 1]$.

As a result, after the expert estimate, for each degree $\beta \in [0, 1]$, we have an interval $\mathbf{x}(\alpha)$ with contains the actual value x with certainty $\alpha = 1 - \beta$. The larger certainty we want, the broader should the corresponding interval be. So, we get a nested family of intervals corresponding to different values α.

An alternative way to describe this nested family of intervals is to describe, for each possible value x of the quantity of interest, the largest possible value α for which this value x belongs to the interval $\mathbf{x}(\alpha)$. This value is usually denoted by $\mu(x)$ and called a *membership function* corresponding to this estimate. Once we know the membership function, we can reconstruct the intervals $\mathbf{x}(\alpha)$ as its α-cuts: $\mathbf{x}(\alpha) = \{x : \mu(x) \ge \alpha\}$; see, e.g., [7], [14].

So, to process expert estimates, we must process the corresponding fuzzy data.

IV. Reliability of interval data

In interval computations, i.e., in processing interval data, we usually assume that all the measuring instruments functioned

correctly, and that all the resulting intervals

$$[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$$

indeed contain the actual value x.

In practice, nothing is 100% reliable. There is a certain probability that a measurement instrument malfunctions. As a result, when we repeatedly measure the same quantity several times, we may have a certain number of measurement results (and hence intervals) which are "way off", i.e., which do not contain the actual value at all.

For example, when we measure the temperature, we will usually get values which are close to the actual temperature, but once in a while the thermometer will not catch the temperature at all, and return a meaningless value like 0. It may be the fault of a sensor, and/or it may be a fault of the processor which processes data from the sensor. Such situations are rare, but when we process a large amount of data, it is typical to encounter some outliers.

Such outliers can ruin the results of data processing. For example, if we compute the average temperature in a given geographic area, then averaging the correct measurement results would lead a good estimate, but if we add an outlier, we can get a nonsense result. For example, based on the measurements of temperature in El Paso in Summer resulting in 95, 100, and 105, we can get a meaningful value

$$\frac{95 + 100 + 105}{3} = 100.$$

However, if we add an outlier 0 to this set of data points, we get a misleading estimate

$$\frac{95 + 100 + 105 + 0}{4} = 75$$

creating the false impression of El Paso climate.

A natural way to characterize the reliability of the data is to set up the bound on the probability p of such outliers. Once we know the value p, then, out of n results of measuring the same quantity, we can dismiss $k \stackrel{\text{def}}{=} p \cdot n$ largest values and k smallest values, and thus make sure that the outliers do not ruin the results of data processing.

V. Need to gauge the reliability of interval data

Where does the estimate p for data reliability come from? The main idea of gauging this value comes from the fact that if we measure the same quantity several times, and all measurements are correct (no outliers), then all resulting intervals $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ contain the same (unknown) value x – and thus, their intersection is non-empty.

If we have an outlier, then it is highly probably that this outlier will be far away from the actual value x – and thus, the intersection of the resulting n intervals (including intervals coming from outliers) will be empty.

In general, if the percentage of outliers does not exceed p, then we expect that out of n given intervals, at least $n - k$ of these intervals (where $k \stackrel{\text{def}}{=} p \cdot n$) correspond to correct measurements and thus, have a non-empty intersection.

So, to check whether our estimate p for reliability is correct, we must be able to check whether out of n given intervals, $n - k$ have a non-empty intersection.

VI. Need to gauge reliability of interval data: multi-D case

In the previous text, we considered a simplified situation in which each measuring instrument measures exactly one quantity. In practice, a measuring instrument often measure several different quantities $x_1, \ldots, x_d$. Due to uncertainty, after the measurement, for each quantity x_i, we have an interval $\mathbf{x}_i$ of possible values. Thus, the set of all possible values of the tuple $x = (x_1, \ldots, x_d)$ is a *box*

$$X = \mathbf{x}_1 \times \ldots \times \mathbf{x}_d = \{(x_1, \ldots, x_d) : x_1 \in \mathbf{x}_1, \ldots, x_d \in \mathbf{x}_d\}.$$

In this multi-D case, if all the measurements are correct (no outliers), all the corresponding boxes $X^{(1)}, \ldots, X^{(n)}$ contain the actual (unknown) tuple and thus, the intersection of all these boxes is non-empty.

Thus, to check whether our estimate p for reliability is correct, we must be able to check whether out of n given boxes, $n - k$ have a non-empty intersection.

VII. How to gauge reliability of fuzzy data

In the fuzzy case, several experts estimate the value of the desired (1-D or multi-D) quantity x. Each of such estimates means that in addition to the (wider) "guaranteed" interval or box $X(0)$ (about which the expert is 100% confident that it contains the actual value of x) we also have narrower intervals (boxes) $X(\alpha)$ which contain x with certainty $1 - \alpha$.

If all experts are right, then at least all the guaranteed boxes $X(0)$ should contain the actual value x. Thus, in this situation, the boxes $X(0)$ corresponding to different experts must have a non-empty intersection. In practice, some experts may be wrong; as a result, the corresponding boxes may be way off, and the intersection of all the experts' boxes may turn out to be empty.

It is reasonable to gauge the reliability of the experts (and, correspondingly, the reliability of the resulting fuzzy data) by the probability p that an expert is wrong. For example, if $p = 0.1$, this means that we expect 90% of the experts to provide us with correct bounds $X(0)$. In this case, we expect that out of all the boxes provided by the experts, we can select 90% of them in such a way that the intersection of these selected boxes will be non-empty.

For boxes $X(\alpha)$ which are known with smaller certainty, the experts themselves agree that these boxes may not cover the actual value x – and thus, the intersection of all such boxes can also turn out to be false. To describe the related reliability, we must know, for every α, the probability p that the corresponding box $X(\alpha)$ does not contain the actual value x. For example, if for $\alpha = 0.5$, we have $p = 0.3$, this means that we expect 70% of the experts' boxes $X(0.5)$ to contain the (unknown) actual value x. In this case, we expect that out of all the boxes $X(0.5)$ based on expert estimates, we can

select 70% of them in such a way that the intersection of these selected boxes will be non-empty.

To check whether the data fits these reliability estimates, we must therefore be able to check whether out of n given boxes, $n-k$ have a non-empty intersection.

VIII. Resulting Computational problem: box intersection problem

Thus, both in the interval and in the fuzzy cases, we need to solve the following computational problem:

- Given:
  - integers d, n, and k; and
  - n d-dimensional boxes
$$X^{(j)} = [\underline{x}_1^{(j)}, \overline{x}_1^{(j)}] \times \ldots \times [\underline{x}_n^{(j)}, \overline{x}_n^{(j)}],$$
$j = 1, \ldots, n$, with rational bounds $\underline{x}_i^{(j)}$ and $\overline{x}_i^{(j)}$.
- *Check:* whether we can select $n-k$ of these n boxes in such a way that the selected boxes have a non-empty intersection.

IX. First result: the box intersection problem is NP-complete

The first result related to this problem is that in general, the above computational problem is NP-hard.

X. The meaning of NP-completeness: a brief explanation

Crudely speaking, NP-completeness means that it is impossible to have an efficient algorithm that solves all particular instances of the above computational problem.

The notion of NP-completeness is relayed to the fact that some algorithms require so much computation time that even for inputs of reasonable size, the required computation time exceeds the lifetime of the Universe – and thus, cannot be practically computed. For example, if for n inputs, the algorithm requires time 2^n, then for $n \approx 300-400$, the resulting computation time is un-realistically large. How can we separate "realistic" ("feasible") algorithms from non-feasible ones?

The running time of an algorithm depends on the size of the input. In the computer, every object is represented as a sequence of bits (0s and 1s). Thus, for every computer-represented object x, it is reasonable to define its *size* (or *length*) $\mathrm{len}(x)$ as the number of bits in this object's computer representation.

It is known that in most feasible algorithms, the running time on an input x is bounded either by the size of the input, or by the square of the size of the input, or, more generally, by a polynomial of the size of the input. It is also known that in most non-feasible algorithms, the running time grows exponentially (or even faster) with the size, so it cannot be bounded by any polynomial. In view of this fact, in theory of computation, an algorithm is usually called feasible if its running time is bounded by a polynomial of the size of the input. This definition is not perfect: e.g., if the running time on input of size n is $10^{40} \cdot n$, then this running time is bounded by a polynomial but it is clearly not feasible. However, this definition is the closest to the intuitive notion of feasible, and thus, the best we have so far.

According to this definition, an algorithm A is called *polynomial time* if there exists a polynomial $P(n)$ such that on every input x, the running time of the algorithm A does not exceed $P(\mathrm{len}(x))$. The class of all the problems which can be solved by polynomial-time algorithms is denoted by P.

What do we mean by "a problem"? In most practical situations, to solve a problem means to find a solution that satisfies some (relatively) easy-to-check constraint: e.g., to design a bridge that can withstand a certain amount of load and wind, to design a spaceship and its trajectory that enables us to deliver a robotic rover to Mars, etc. In all these cases, once we have a candidate for a solution, we can check, in reasonable (polynomial) time whether this candidate is indeed a solution. In other words, once we guessed a solution, we can check its correctness in polynomial time. In theory of computation, this procedure of guess-then-compute is called *non-deterministic computation*, so the class of all problems for which solution can be checked in polynomial time is called Non-deterministic Polynomial, or NP, for short.

Most computer scientists believe that not all problems from the class NP can be solved in polynomial time, i.e., that NP$\neq P$. However, no one has so far been able to prove that this belief is indeed true. What is known is that some problems from the class NP are the hardest in this class – in the sense that every other problem from the class NP can be reduced to such a problem.

Specifically, a general problem (not necessarily from the class NP) is called *NP-hard* if every problem from the class NP can be reduced to particular cases of this problem. If a problem from the class NP is NP-hard, we say that it is *NP-complete*.

One of the best known examples of NP-complete problems is the problem of *propositional satisfiability* for formulas in 3-Conjunctive Normal Form (3-CNF). Let us describe this problem is some detail. We start with v Boolean variables $z_1, \ldots, z_v$, i.e., variables which can take only values "true" or "false". A *literal* ℓ is defined as a variable z_i or its negation $\neg z_i$. A *clause* is defined as a formula of the type $\ell_1 \vee \ell_2 \vee \ldots \vee \ell_m$. Finally, a *propositional formula in Conjunctive Normal Form (CNF)* is defined as a formula F of the type $C_1 \,\&\, \ldots \,\&\, C_n$, where $C_1, \ldots, C_n$ are clauses. This formula is called a *3-CNF* formula if every clause has at most 3 literals, and a *2-CNF* formula if every clause has at most 2 literals.

The propositional satisfiability problem is as follows:

- Given a propositional formula F (e.g., a formula in CNF);
- Find the values of the variables $z_1, \ldots, z_v$ which make the formula F true.

For the propositional satisfiability problem, the proof of NP-hardness is somewhat complex. However, once this NP-hardness is proven, we can prove the NP-hardness of other problems by reducing satisfiability to these problems.

Indeed, by definition, NP-hardness of satisfiability means that every problem from the class NP can be reduced to satisfiability. If we can reduce satisfiability to some other problem, this means that by combining these two reductions, we can reduce every problem from the class NP to this new problem – and thus, that this new problem is also NP-hard.

For a more detailed and more formal definition of NP-hardness, see, e.g., [10], [15].

XI. The box intersection problem is NP-complete

In the Appendix, we present a proof that the box intersection problem is NP-complete. The main ideas of this proof comes from [9].

XII. Case of fixed dimension: efficient algorithm for gauging reliability

In general, when we allow unlimited dimension d, the box intersection problem (computational problem related to gauging reliability) is computationally difficult (NP-hard).

In practice, however, the number d of quantities measured by a sensor is small: e.g.,

- a GPS sensor measures 3 spatial coordinates;
- a weather sensor measures (at most) 5: temperature, atmospheric pressure, and the 3 dimensions of the wind vector, etc.

It turns out that if we limit ourselves to the case of a fixed dimension d, then we can solve the above computational problem in polynomial time $O(n^d)$; see, e.g., [3].

Indeed, for each of d dimensions x_i ($1 \le i \le d$), the corresponding n intervals have $2n$ endpoints $\underline{x}_i^{(j)}$ and $\overline{x}_i^{(j)}$. Let us show if there exists a vector x which belongs to $\ge n-k$ boxes $X^{(j)}$, then there also exists another point y with this property in which every coordinate y_i coincides with one of the endpoints. Indeed, if for some i, the value x_i is not an endpoint, then we can take the closest endpoint as y_i. One can easily check that this change will keep the vector is all the boxes $X^{(j)}$.

Thus, to check whether there exists a vector x that belongs to at least $n-k$ boxes $X^{(j)}$, it is sufficient to check whether there exist a vector formed by endpoints which satisfies this property. For each vector $y = (y_1, \ldots, y_d)$ and for each box $X^{(j)}$, it takes $d = O(1)$ steps to check whether $y \in X^{(j)}$. After repeating this check for all n boxes, we thus check whether this vector y satisfies the desired property in time $n \cdot O(1) = O(n)$.

For each of d dimensions, there are $2n$ possible endpoints; thus, there are $(2n)^d$ possible vectors y formed by such endpoints. For each of these vectors, we need time $O(n)$, so the overall computation time for this procedure requires time $O(n) \cdot (2n)^d = O(n^{d+1})$ – i.e., indeed time which grows polynomially with n.

XIII. Remaining problem

In the previous section, we have shown that for a bounded dimension d, we can solve the box intersection problem in polynomial time. However, as we have mentioned, polynomial time does not always mean that the algorithm is practically feasible.

For example, for a meteorological sensor, the dimension d is equal to 5, so we need n^6 computational steps. For $n = 10$, we get 10^6 steps, which is easy to perform. For $n = 100$, we need $100^6 = 10^{12}$ steps which is also doable – especially on a fast computer. However, for a very reasonable amount of $n = 10^3 = 1000$ data points, the above algorithm requires $1000^6 = 10^{18}$ computational steps – which already requires a long time, and for $n = 10^4$ data points, the algorithm requires a currently practically impossible amount of 10^{24} computational steps.

It is therefore desirable to speed up the computations. In this paper, we show that we can achieve a significant speed up if we use quantum computations.

XIV. Quantum computations: a reminder

Before we explain how exactly quantum computations can speed up the computations needed to gauge reliability, let us briefly recall how quantum effects can be used to speed up computations.

In this paper, we will use Grover's algorithm for quantum search. 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., [4], [5], [13]) can find this element much faster – in $O(\sqrt{N})$ time.

Specifically, Grover's 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, with probability $\ge 1-\delta$, 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).

For the Grover's algorithm, 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.

Brassard et al. used the ideas behind Grover's algorithm to produce a new quantum algorithm for *quantum counting*; see, e.g., [2], [13]. 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.

XV. Quantum computations can drastically speed up gauging reliability

As a part of the above algorithm for checking box intersections, we search among $O(n^d)$ vectors y for a vector that belongs to at least $n-k$ boxes $X^{(j)}$. For each of these vectors y, we need to find to how many of n boxes $X^{(j)}$ the vector y belongs; this requires time $O(n)$.

For each vector y, we can use the quantum counting algorithm to compute the number of boxes in time $O(\sqrt{n})$. We can then use Grover's algorithm to reduce the non-quantum search of $N = O(n^d)$ vectors to a search whose time is equivalent to processing $\sqrt{N} = O(n^{d/2})$ such vectors. For each of these vectors, we need time $O(\sqrt{n})$. Thus, if we use quantum computations, we need the total computation time $O(n^{d/2}) \cdot O(\sqrt{n}) = O(n^{(d+1)/2})$.

This time is much smaller than the non-quantum computation time $O(n^{d+1})$. For example, for the above meteorological example of $n = 10^4$ and $d = 5$, the non-quantum algorithm requires a currently impossible amount of 10^{24} computational steps, while the quantum algorithm requires only a reasonable amount of 10^{12} steps.

Comment. A similar square root reduction can be achieved in the general case, but for general d, $n^{(d+1)/2}$ computational steps may still take too long.

XVI. Conclusion

In traditional interval computations, we assume that the interval data corresponds to guaranteed interval bounds, and that fuzzy estimates provided by experts are correct. In practice, measuring instruments are not 100% reliable, and experts are not 100% reliable, we may have estimates which are "way off", intervals which do not contain the actual values at all. Usually, we know the percentage of such outlier un-reliable measurements. It is desirable to check that the reliability of the actual data is indeed within the given percentage. In this paper, we have shown that:

- in general, the problem of checking (gauging) this reliability is computationally intractable (NP-hard);
- in the reasonable case when each sensor measures a small number of different quantities, it is possible to solve this problem in polynomial time; and
- quantum computations can drastically reduce the required computation time.

Acknowledgments

This work was supported by the Alliances for Graduate Education and the Professoriate (AGEP) grant HRD-0302788 from the National Science Foundation (NSF).

The authors are thankful to Gilles Chabert, Alexandre Goldsztejn, Luc Jaulin, Vladik Kreinovich, and Alasdair Urquhart for their help and encouragement, and to the anonymous referees for valuable suggestions.

References

[1] G. Ausiello, P. Crescenzi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, Springer-Verlag, Berlin-Heidelberg, 1999.

[2] 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.

[3] A. Goldsztejn, private communication, 2007.

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

[5] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack", *Phys. Rev. Lett.*, 1997, Vol. 79, No. 2, pp. 325–328.

[6] 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.

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

[8] R. Kohli, R. Krishnamurthi, and P. Mirchandani, "The Minimum Satisfiability Problem", *SIAM Journal on Discrete Mathematics*, 1994, Vol. 7, No. 2, pp. 275–283.

[9] V. Kreinovich, private communication, 2007.

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

[11] V. Kreinovich and L. Longpré, "Fast Quantum Algorithms for Handling Probabilistic and Interval Uncertainty", *Mathematical Logic Quarterly*, 2004, Vol. 50, No. 4/5, pp. 507–518.

[12] M. Martinez, L. Longpré, V. Kreinovich, S. A. Starks, and H. T. Nguyen, "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, pp. 395–400.

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

[14] H. T. Nguyen and E. A. Walker, *First Course in Fuzzy Logic*, CRC Press, Boca Raton, FL, 2006.

[15] C. H. Papadimitriou, *Computational Complexity*, Addison Wesley, San Diego, 1994.

[16] S. Rabinovich, *Measurement Errors: Theory and Practice*, American Institute of Physics, N.Y., 2005.

Appendix

As we have mentioned in the main text, in gauging reliability, it is important to be able to solve the following box intersection problem:

- Given: a set of n d-dimensional boxes, and a number $k < n$.
- Check: is there a vector x which belongs to at least $n-k$ of these n boxes?

This box intersection problem obviously in NP: it is easy to check that a given vector x belongs to each of the boxes, and thus, to check whether it belongs to at least $n-k$ of the boxes. So we only need a proof of NP-hardness.

The proof is by reduction from the following auxiliary "limited clauses" problem which has been proved to be NP-complete:

- Given: a 2-CNF formula F and a number k,
- check: is there a Boolean vector which satisfies at most k clauses of F.

This problem was proved to be NP-complete in [8] (see also [1], p. 456).

As we have mentioned in the main text of this paper, to prove the NP-hardness of our box intersection problem, it is therefore sufficient to be able to reduce this "limited clauses" problem to the box intersection problem.

Indeed, suppose that we are given a 2-CNF formula F. Let us denote the number of Boolean variables in this formula by d, and the overall number of clauses in this formula F by n. Based on the formula F, let us build a set of n d-dimensional boxes, one for each clause. If clause C_i contains Boolean variables z_{i1} and z_{i2} variables, then the i-th box $X^{(i)}$ has sides $[0,1]$ in all dimensions except in the dimensions associated with variables z_{i1} and z_{i2}. For those two dimensions, the side is:

- $[0,0]$ if the variable occurs positively in the clause (i.e., if the clause contains the positive literal z_{ij}), and
- $[1,1]$ is the variable occurs negatively in the clause (i.e., if the clause contains the negative literal $\neg z_{ij}$).

According to the construction:

- for a clause $z_{i1} \vee z_{i2}$, a vector x belongs to the box

$$X^{(i)} = \ldots \times [0,1] \times [0,0] \times [0,1] \times \ldots \times [0,1] \times [0,0] \times [0,1] \times \ldots$$

  if and only of $x_{i1} = 0$ and $x_{i2} = 0$;
- for a clause $z_{i1} \vee \neg z_{i2}$, a vector x belongs to the box $X^{(i)}$ if and only of $x_{i1} = 0$ and $x_{i2} = 1$;
- for a clause $\neg z_{i1} \vee z_{i2}$, a vector x belongs to the box $X^{(i)}$ if and only of $x_{i1} = 1$ and $x_{i2} = 0$;
- for a clause $\neg z_{i1} \vee \neg z_{i2}$, a vector x belongs to the box $X^{(i)}$ if and only of $x_{i1} = 1$ and $x_{i2} = 1$.

The claim is that there exists a vector x which belongs to at least $n-k$ of these n boxes if and only if there is a Boolean vector z which satisfies at most k clauses of the formula F.

Suppose that there exists a vector x which belongs to at least $n-k$ of these n boxes. According to our construction, each box $X^{(i)}$ comes from a clause C_i that contains variables z_{i1} and z_{i2}. For each box $X^{(i)}$ to which the vector x belongs, make $z_{i1} =$"false" if the box has $[0,0]$ on the side associated with variable z_{i1}. Similarly, we make $z_{i2} =$"false" if the box has $[0,0]$ on the side associated with variable z_{i2}. Because of the way the boxes were build, the Boolean vector we build will make the clause associated with the box corresponding box $X^{(i)}$ false.

For example, if the clause is $z_{i1} \vee z_{i2}$, then the box will have $[0,0]$ for the sides associated with both variable, so they will be both assigned the "false" Boolean value, making the clause false. This means that the Boolean formula built will make at least $n-k$ clauses become false. This formula will satisfy at most $k = n-(n-k)$ clauses.

In the opposite direction, if there is a Boolean vector z which satisfies at most k clauses of the formula F, build a vector $x = (x_1, \ldots, x_n)$ which has value:

- $x_i = 0$ in dimension i if the Boolean variable z_i associated with this dimension is false, and
- $x_i = 1$ otherwise.

One can check that for this arrangement, $x \in X^{(i)}$ if and only if the original Boolean vector z made the corresponding clause C_i false.

Since the Boolean vector z satisfies at most k clauses of the formula F, it makes at least $n-k$ clauses false. This means that the vector x that we have built will belong to all the boxes associated with at least $n-k$ clauses that are false.

The reduction is proven, and so is NP-hardness.