From Single to Double Use Expressions, with Applications to Parametric Interval Linear Systems: On Computational Complexity of Fuzzy and Interval Computations

Joe Lorkowski
Department of Computer Science
University of Texas at El Paso
500 W. University
El Paso, Texas 79968, USA
Email: lorkowski@ieee.org

***Abstract*—In many practical problems, we need to estimate the range of a given expression $f(x_1,\ldots,x_n)$ when each input x_i belongs to a known interval $[\underline{x}_i,\overline{x}_i]$ – or when each input x_i is described by a known fuzzy set. It is known that this problem is easy to solve when we have a Single Use Expression, i.e., an expression in which each variable x_i occurs only once. In this paper, we show that for similarly defined Double Use Expressions, the corresponding range estimation problem is NP-hard. Similar problems are analyzed for the problem of solving linear systems under interval (and fuzzy) uncertainty.**

I. Formulation of the Problem

Need for data processing. In many real-life situations, we need to process data, i.e., use the estimated values $\widetilde{x}_1,\ldots,\widetilde{x}_n$ to estimate the value $\widetilde{y}$ of another quantity.

This may happen because we are interested in the value of a quantity that is difficult or even impossible to measure directly – e.g., the amount of oil in a well or the distance to a faraway star – but which can be estimated based on some related easier-to-measure quantities (e.g., the angles to the star from two different telescopes).

The need for data processing also emerges we try to predict the future values of some quantities based on the their current values and the known dynamical equations. An example of such situation is when we want to predict tomorrow's weather based on today's meteorological observations.

In all these cases, we apply an appropriate algorithm f to the known estimates and get the desired estimate $\widetilde{y} = f(\widetilde{x}_1,\ldots,\widetilde{x}_n)$. This algorithm can be as simple as applying an explicit formula (to find the distance to a star) or as complex as solving a system of partial differential equations (to predict weather).

Need to take uncertainty into account when processing data. Estimates are never absolutely accurate: for each of the input quantities the estimate $\widetilde{x}_i$ is, in general, different from its actual (unknown) value x_i. As a result, even if the algorithm f is exact – i.e., it would have produced the exact value $y = f(x_1,\ldots,x_n)$ if we plug in the exact values x_i – because of the uncertainty $\widetilde{x}_i \neq x_i$, the value $\widetilde{y}$ is, in general, different from the desired value y.

It is therefore necessary to analyze how the uncertainty in estimating x_i affects the uncertainty with which we determine y.

Need for interval data processing and interval computations. When estimates come from measurements, the difference $\Delta_i = \widetilde{x}_i - x_i$ is called a *measurement error*.

Sometimes, we know the probabilities of different values of measurement errors, but often, the only information that we have about the measurement error Δx_i is the upper bound Δ_i provided by the manufacturer: $|\Delta x_i| \le \Delta_i$; see, e.g., [14]. In such situations, the only information that we have about x_i is that x_i belongs to the interval $\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$.

Different values x_i from these intervals $\mathbf{x}_i$ lead, in general, to different values $y = f(x_1,\ldots,x_n)$. So, to gauge the uncertainty in y, it is necessary to find the range of all possible values of y:

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

This range is usually denoted by $f(\mathbf{x}_1,\ldots,\mathbf{x}_n)$.

The problem of estimating this range based on given intervals $\mathbf{x}_i$ constitutes the main problem of *interval computations*; see e.g., [7], [10].

Need for fuzzy data processing. In many practical situations, estimates $\widetilde{x}_i$ come from experts. In this case, we do not have guaranteed upper bounds on the estimation error $\Delta x_i = \widetilde{x}_i - x_i$. Instead, we have expert estimates of their accuracy – estimates formulated in terms of words from natural language such as "approximately 0.1". One of the main ways to formalize such informal ("fuzzy") statements is to use *fuzzy logic* (see, e.g., [8], [11]), techniques specifically designed for the purpose of such formalization.

In fuzzy logic, to describe a fuzzy property $P(x)$ of real numbers (such as "approximately 0.1"), we assign, to every real number x, the degree $\mu_P(x) \in [0,1]$ which, according to an expert, the number x satisfies this property.

- If the expert is absolutely sure, this degree is 1.
- Otherwise, the degree takes value between 0 and 1.

Once we know the experts' degrees d_1, d_2, $\ldots$, of different statements S_1, S_2, $\ldots$, we need to estimate the degree d to which a logical combination like $S_1 \vee S_2$ or $S_1 \,\&\, S_2$ hold. In other words, for each pair of values d_1 and d_2, we must select:

- an estimate for $S_1 \vee S_2$ – which will be denoted by $f_\vee(d_1, d_2)$,
- an estimate for $S_1 \,\&\, S_2$ - which will be denoted by $f_\&(d_1, d_2)$,
- etc.

Natural requirements – e.g.,

- that $S \,\&\, S$ mean the same as S,
- that $S_1 \,\&\, S_2$ means the same as $S_2 \,\&\, S_1$,

etc. – uniquely determine operations $f_\&(d_1, d_2) = \min(d_1, d_2)$ and $f_\vee(d_1, d_2) = \max(d_1, d_2)$ [8], [11].

A real number y is a possible value of the desired quantity if and only if there exist values $x_1, \ldots, x_n$ which are possible values of the input quantities and for which $y = f(x_1, \ldots, x_n)$:

$$y \text{ is possible } \Leftrightarrow$$

$$\exists x_1 \ldots \exists x_n \Big((x_1 \text{ is possible}) \,\&\, \ldots \,\&\, (x_n \text{ is possible}) \,\&$$

$$y = f(x_1, \ldots, x_n) \Big).$$

Once we know the degrees $\mu_i(x_i)$ corresponding to the statements "x_i is possible", we can then apply the above "and" and "or" operations $f_\&(d_1, d_2) = \min(d_1, d_2)$ and $f_\vee(d_1, d_2) = \max(d_1, d_2)$ (and the fact that an existential quantifier $\exists$ is, in effect, an infinite "or") to estimate the degree $\mu(y)$ to which y is possible:

$$\mu(y) = \max\{\min(\mu_1(x_1), \ldots, \mu_n(x_n) : y = f(x_1, \ldots, x_n)\}.$$

This formula was first proposed by Zadeh, the father of fuzzy logic, and is usually called *Zadeh's extension principle*.

From the computational viewpoint, fuzzy data processing can be reduced to interval data processing. An alternative way to describe a membership function $\mu_i(x_i)$ is to describe, for each possible values $\alpha \in [0,1]$, the set of all values x_i for which the degree of possibility is at least α. This set

$$\{x_i : \mu_i(x_i) \ge \alpha\}$$

is called an *alpha-cut* and is denoted by $X_i(\alpha)$.

It is known (see, e.g., [8], [11]), that for alpha-cuts, Zadeh's extension principle takes the following form: for every α, we have

$$Y(\alpha) = \{f(x_1, \ldots, x_n) : x_i \in X_i(\alpha)\}.$$

Thus, for every α, finding the alpha-cut of the resulting membership function $\mu(y)$ is equivalent to applying interval computations to the corresponding intervals $X_1(\alpha)$, $\ldots$, $X_n(\alpha)$.

Because of this reduction, in the following text, we will only consider the case of interval uncertainty.

In general, interval computations are NP-hard. In general, the main problem of interval computations is NP-hard – meaning that, if (as most computer scientists believe) P≠NP, no algorithm can always compute the desired range in feasible time (i.e., in time which is bounded by the polynomial of the length of the input).

Thus, every feasible algorithm for estimating the range $\mathbf{y}$ sometimes leads to an over- (or under-) estimation.

Comment. NP-hardness of interval computations was first proven in [4], [5] by reducing, to this problem, a known NP-hard problem of propositional satisfiability (SAT) for propositional formulas in Conjunctive Normal Form (CNF): given an expression of the type

$$(v_1 \vee \neg v_2 \vee v_3) \,\&\, (v_1 \vee \neg v_4) \,\&\, \ldots,$$

check whether there exist Boolean (true-false) values v_i that make this formula true.

The disjunctions $(v_1 \vee \neg v_2 \vee v_3)$, $(v_1 \vee \neg v_4)$, $\ldots$, are called *clauses*, and variables and their negations are called *literals*.

An overview of related NP-hardness results is given in [9]. Later papers showed that many simple interval computation problems are NP-hard: e.g., the problem of computing the range of sample variance

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

where $E = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$; see, e.g., [2], [3].

Naive (straightforward) interval computations. Historically the first algorithm for estimating the range consists of the following. For each elementary arithmetic operation $\oplus$ like addition or multiplication, due to monotonicity, we can explicitly describe the corresponding range $\mathbf{x}_1 \oplus \mathbf{x}_2$:

$$[\underline{x}_1, \overline{x}_1] + [\underline{x}_2, \overline{x}_2] = [\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2];$$

$$[\underline{x}_1, \overline{x}_1] - [\underline{x}_2, \overline{x}_2] = [\underline{x}_1 - \overline{x}_2, \overline{x}_1 - \underline{x}_2];$$

$$[\underline{x}_1, \overline{x}_1] \cdot [\underline{x}_2, \overline{x}_2] = [\min(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2),$$

$$\max(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2)];$$

$$\frac{1}{[\underline{x}_1, \overline{x}_1]} = \left[\frac{1}{\overline{x}_1}, \frac{1}{\underline{x}_1}\right] \text{ if } 0 \notin [\underline{x}_1, \overline{x}_1];$$

$$\frac{[\underline{x}_1, \overline{x}_1]}{[\underline{x}_2, \overline{x}_2]} = [\underline{x}_1, \overline{x}_1] \cdot \frac{1}{[\underline{x}_2, \overline{x}_2]}.$$

These formulas form *interval arithmetic*.

To estimate the range, we:

- *parse* the original algorithm f – i.e., represent it as a sequence of elementary arithmetic operations, and then
- replace each operation with numbers with the corresponding operation with intervals.

Sometimes we thus get the exact range, but sometimes, we only get an *enclosure* – i.e., an interval that contains the exact range but is different from it. For example, for a function $f(x_1) = x_1 \cdot (1 - x_1)$ on the interval $\mathbf{x}_1 = [0, 1]$, the actual range is $[0, 0.25]$, but naive interval computations return an enclosure. Specifically, the original algorithm can be described as the sequence of the following two steps:

$$r_1 = 1 - x_1; \quad y = x_1 \cdot r_1.$$

Thus, the resulting naive interval computations lead to

$$\mathbf{r}_1 = [1, 1] - [0, 1] = [1 - 1, 1 - 0] = [0, 1];$$

$$\mathbf{y} = [0, 1] \cdot [0, 1] =$$

$$[\min(0 \cdot 0, 0 \cdot 1, 1 \cdot 0, 1 \cdot 1), \max(0 \cdot 0, 0 \cdot 1, 1 \cdot 0, 1 \cdot 1)] =$$

$$[0, 1].$$

Comment. It should be mentioned there exist more sophisticated algorithms for computing the interval range, algorithms that produce much more accurate estimation for the ranges, and these algorithms form the bulk of interval computations results [7], [10].

Single Use Expressions. There is a known case when naive interval computations lead to the exact range – case of *Single Use Expressions* (SUE), i.e., expressions $f(x_1, \ldots, x_n)$ in which each variable occurs only once; see, e.g., [6], [7], [10].

For example, $x_1 \cdot x_2 + x_3$ is a SUE, while the above example $x_1 \cdot (1 - x_1)$ is not, because in this expression, the variable x_1 occurs twice.

A natural open problem. What if we have double-use expressions, i.e., expressions in which each variable occurs at most twice? Is it possible to always compute the range of such expressions in feasible time? If yes, then what about triple-use expressions?

These are the questions that we answer in this paper.

II. Analysis of the Problem and the Main Result

Since the original proof of NP-hardness of interval computations comes from reduction to SAT, let us consider the corresponding SAT problems. Namely, we will say that

- a propositional formula of the above type is a Single-Use Expression (SUE) if in this formula, each Boolean variable occurs only once; and
- a Double-Use Expression (DUE) if each Boolean variable occurs at most twice.

For example:

- $(v_1 \vee \neg v_2 \vee v_3) \,\&\, (v_4 \,\&\, v_5)$ is a SUE formula, and
- $(v_1 \vee \neg v_2 \vee v_3) \,\&\, (v_1 \,\&\, \neg v_3)$ is a DUE formula: here v_1 and v_3 occur twice, and v_2 occurs once.

For propositional formulas, checking satisfiability of SUE formulas is feasible:

Proposition 1. *There exists a feasible algorithm for checking propositional satisfiability of SUE formulas.*

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

For DUE formulas, we have a similar result:

Proposition 2. *There exists a feasible algorithm for checking propositional satisfiability of DUE formulas.*

One may thus expect that the interval computations problem for DUE expressions is also feasible. However, our result is opposite:

Proposition 3. *The main problem of interval computations for DUE formulas is NP-hard.*

III. Beyond Range Estimation: Solving Interval Linear Systems

Systems of interval linear equations: reminder. In many cases, instead of a known algorithm, we only have implicit relations between the inputs x_i and the desired value y. The simplest such case is when these relations are linear, i.e., when we need to determine the desired values $y_1, \ldots, y_n$ from the system of equations

$$\sum_{j=1}^{n} a_{ij} \cdot y_j = b_i,$$

where we know estimates for a_{ij} and b_i – e.g., intervals $\mathbf{a}_{ij}$ and $\mathbf{b}_i$ of possible values of these variables.

In this case, a natural question is to find the range of all possible values y_j when a_{ij} takes values from $\mathbf{a}_{ij}$ and b_i takes values from the interval $\mathbf{b}_i$.

Systems of interval linear equations: what is known about their computational complexity. It is known that computing the desired ranges is an NP-hard problem; see, e.g., [9].

However, a related problem is feasible: given a sequence of values $x_1, \ldots, x_n$ check whether there exist values $a_{ij} \in \mathbf{a}_{ij}$ and $b_i \in \mathbf{b}_i$ for which the above system is true.

This algorithm can be easily described in SUE terms: for every i, the expression $\sum\limits_{j=1}^{n} a_{ij} \cdot y_j$ is a SUE, thus, its range can be found by using naive interval computation, as $\sum\limits_{j=1}^{n} \mathbf{a}_{ij} \cdot y_j$. The above equality is possible if and only if this range and the interval $\mathbf{b}_i$ have a non-empty intersection for every i:

$$\left(\sum_{j=1}^{n} \mathbf{a}_{ij} \cdot y_j \right) \cap \mathbf{b}_i \neq \emptyset.$$

Checking whether two intervals $[\underline{x}_1, \overline{x}_1]$ and $[\underline{x}_2, \overline{x}_2]$ have a non-empty intersection is easy:

$$[\underline{x}_1, \overline{x}_1] \cap [\underline{x}_2, \overline{x}_2] \neq \emptyset \Leftrightarrow \underline{x}_1 \le \overline{x}_2 \,\&\, \underline{x}_2 \le \overline{x}_1.$$

Thus, we indeed have a feasible algorithm; this criterion is known as the Oettli-Prager criterion [7], [10].

Parametric interval linear systems: reminder. In some cases, we have additional constraints on the values a_{ij}. For example, we may know that the matrix a_{ij} is symmetric:

$a_{ij} = a_{ji}$. In this case, not all possible combinations $a_{ij} \in \mathbf{a}_{ij}$ are allowed: only those for which $a_{ij} = a_{ji}$. In this case, it is sufficient to describe the values a_{ij} for $i \le j$, the others can be expressed in terms of these ones.

In general, we can consider a *parametric* system in which we have k parameters $p_1, \ldots, p_k$ that take values from known intervals $\mathbf{p}_1, \ldots, \mathbf{p}_k$, and values a_{ij} and b_i are linear functions of these variables: $a_{ij} = \sum\limits_{\ell=1}^{k} a_{ij\ell} \cdot p_\ell$ and $b_i = \sum\limits_{\ell=1}^{k} b_{i\ell} \cdot p_\ell$, for known coefficients $a_{ij\ell}$ and $b_{i\ell}$.

Parametric interval linear systems: what is known about their computational complexity. This problem is more general than the above problem of solving systems of linear equations. Thus, since the above problem is NP-hard, this problem is NP-hard as well.

The next natural question is: is it possible to check whether a given tuple $x = (x_1, \ldots, x_n)$ is a solution to a given parametric interval linear system, i.e., whether there exist values p_ℓ for which $\sum\limits_{j=1}^{n} a_{ij} \cdot y_j = b_i$.

The first result of this type was proven in [12], [13]. In this paper, it is shown that if each parameter p_i occurs only in one equation (even if it occurs several times in this equation), then checking is still feasible.

The proof can also be reduced to the SUE case: indeed, in this case, it is sufficient to consider one equation at a time – since no two equation share a parameter. For each i, the corresponding equation $\sum\limits_{j=1}^{n} a_{ij} \cdot y_j = b_i$ takes the form

$$\sum_{j=1}^{n} \sum_{\ell=1}^{k} a_{ij\ell} \cdot y_j \cdot p_\ell = \sum_{\ell=1}^{k} b_{i\ell} \cdot p_\ell,$$

i.e., the (SUE) linear form

$$\sum_{\ell=1}^{k} A_{i\ell} \cdot p_\ell = 0,$$

where

$$A_{i\ell} = \sum_{j=1}^{n} a_{ij\ell} \cdot y_j - b_{i\ell},$$

and we already know that checking the solvability of such an equation is feasible.

Natural questions. What happens if we we allow each parameter to occur several times? What if each parameter occurs only in one equation, but the dependence of a_{ij} and b_i on the parameters can be quadratic (this question was asked by G. Alefeld):

$$a_{ij} = a_{ij0} + \sum_{\ell=1}^{k} a_{ij\ell} \cdot p_\ell + \sum_{\ell=1}^{k} \sum_{\ell'=1}^{k} a_{ij\ell\ell'} \cdot p_\ell \cdot p_{\ell'};$$

$$b_i = b_{i0} + \sum_{\ell=1}^{k} b_{i\ell} \cdot p_\ell + \sum_{\ell=1}^{k} \sum_{\ell=1}^{k} b_{i\ell\ell'} \cdot p_\ell \cdot p_{\ell'}.$$

In this paper, we provide answers to both questions.

Proposition 4. *When we only allow linear dependence on parameters, there exists a feasible algorithm that checks whether a given tuple x belongs to a solution set of a parametric interval linear system.*

Proposition 5. *For parametric interval linear systems with quadratic dependence on parameters, the problem of checking whether a given tuple x belongs to a solution set of a given system is NP-hard even if we only consider systems in which each parameter occurs only on one equation.*

IV. Proofs

Proof of Proposition 1. This algorithm is simple because every SUE propositional formula is satisfiable. Indeed, each variable v_i occurs only once.

- If it occurs as negation $\neg v_i$, we set v_i to false, then $\neg v_i$ is true.
- If it occurs without negation, then we set v_i to be true.

In both cases, for this choice, all the literals v_i or $\neg v_i$ are true, and thus, the whole formula is true.

Proof of Proposition 2.

1°. Let us show that we can "eliminate" each variable v_i – i.e., in feasible time, reduce the problem of checking satisfiability of the original formula to the problem of checking satisfiability of a formula of the same (or smaller) length, but with one fewer variable.

In this proof, we will use the two facts:

- that if a formula $A \,\&\, B$ is true, then both A and B are true; and
- that if a formula A is true, then the formula $A \vee B$ is also true.

2°. Since the formula is DUE, each variable v_i occurs at most twice, i.e., it occurs either once or twice. We will consider these two cases one by one.

3°. Let us first consider the case when the formula occurs once. In this case, it occurs either as v_i or as $\neg v_i$. Let us consider these two situations one by one.

3.1°. If the variable v_i occurs only once as $\neg v_i$, then the formula has the form $(\neg v_i \vee r) \,\&\, R$, where:

- r denotes the remaining part of the clause containing $\neg v_i$, and
- R is the conjunction of all the other clauses.

Let us show that the satisfiability of the original formula is equivalent to satisfiability of a shorter formula R that does not contain v_i at all. Indeed:

- If the original formula $(\neg v_i \vee r) \,\&\, R$ is satisfied, this means that it is true for some selection of variables. For this same selection of variables, due to the above fact about &, the formula R is true as well, so the formula R is also satisfied.
- Vice versa, let us assume that R is satisfied. This means that for some selection of variables, R is true. If we now take v_i to be false, then $\neg v_i$ is true and thus (due to the

above fact about $\vee$), the clause $(\neg v_i \vee r)$ will be true as well. Thus, the whole formula $(\neg v_i \vee r) \,\&\, R$ will be true.

3.2°. Similarly, if the variable v_i occurs only once as v_i, then the formula has the form $(v_i \vee r) \,\&\, R$, where:

- r denotes the remaining part of the clause containing v_i, and
- R is the conjunction of all the other clauses.

Let us show that the satisfiability of the original formula is equivalent to satisfiability of a shorter formula R that does not contain v_i at all. Indeed:

- If the original formula $(v_i \vee r) \,\&\, R$ is satisfied, this means that it is true for some selection of variables. For this same selection of variables, due to the above fact about &, the formula R is true as well, so the formula R is also satisfied.
- Vice versa, let us assume that R is satisfied. This means that for some selection of variables, R is true. If we now take v_i to be true; then (due to the above fact about $\vee$), the clause $(v_i \vee r)$ will be true as well. Thus, the whole formula $(v_i \vee r) \,\&\, R$ will be true.

4°. If the variable v_i occurs twice, we have three possibilities:

- it occurs both times as v_i;
- it occurs both times as $\neg v_i$; and
- it occurs once as v_i and once as $\neg v_i$.

Let us consider these three possibilities one by one.

4.1°. If the variable v_i occurs twice as v_i, then the formula has the form $(v_i \vee r) \,\&\, (v_i \vee r') \,\&\, R$, where:

- r denotes the remaining part of the first clause containing v_i,
- r' denotes the remaining part of the second clause containing v_i, and
- R is the conjunction of all the other clauses.

Let us show that the satisfiability of the original formula is equivalent to satisfiability of a shorter formula R that does not contain v_i at all. Indeed:

- If the original formula $(v_i \vee r) \,\&\, (v_i \vee r') \,\&\, R$ is satisfied, this means that it is true for some selection of variables. For this same selection of variables, due to the above fact about &, the formula R is true as well, so the formula R is also satisfied.
- Vice versa, let us assume that R is satisfied. This means that for some selection of variables, R is true. If we now take v_i to be true; then (due to the above fact about $\vee$), the clauses $(v_i \vee r)$ and $(v_i \vee r')$ will be true as well. Thus, the whole formula $(v_i \vee r) \,\&\, (v_i \vee r') \,\&\, R$ will be true.

4.2°. If the variable v_i occurs twice as $\neg v_i$, then the formula has the form $(\neg v_i \vee r) \,\&\, (\neg v_i \vee r') \,\&\, R$, where:

- r denotes the remaining part of the first clause containing $\neg v_i$,
- r' denotes the remaining part of the second clause containing $\neg v_i$, and
- R is the conjunction of all the other clauses.

Let us show that the satisfiability of the original formula is equivalent to satisfiability of a shorter formula R that does not contain v_i at all. Indeed:

- If the original formula $(\neg v_i \vee r) \,\&\, (\neg v_i \vee r') \,\&\, R$ is satisfied, this means that it is true for some selection of variables. For this same selection of variables, due to the above fact about &, the formula R is true as well, so the formula R is also satisfied.
- Vice versa, let us assume that R is satisfied. This means that for some selection of variables, R is true. If we now take v_i to be false; then $\neg v_i$ is true, so (due to the above fact about $\vee$), the clauses $(\neg v_i \vee r)$ and $(\neg v_i \vee r')$ will be true as well. Thus, the whole formula
$$(\neg v_i \vee r) \,\&\, (\neg v_i \vee r') \,\&\, R$$
is true.

4.3°. Finally, if the variable v_i occurs once as v_i and once as $\neg v_i$, then the formula has the form $(v_i \vee r) \,\&\, (\neg v_i \vee r') \,\&\, R$, where:

- r denotes the remaining part of the clause containing v_i,
- r' denotes the remaining part of the clause containing $\neg v_i$, and
- R is the conjunction of all the other clauses.

Let us show that the satisfiability of the original formula is equivalent to satisfiability of a shorter formula $(r \vee r') \,\&\, R$ that does not contain v_i at all (this fact is known as *resolution rule*). Indeed:

- If the original formula $(v_i \vee r) \,\&\, (\neg v_i \vee r') \,\&\, R$ is satisfied, this means that that for some combination of variables, all three formulas R, $v_i \vee r$, and $\neg v_i \vee r'$ are true. In this combination, v_i is either true or false.
  - If v_i is true, then from the fact that $\neg v_i \vee r'$ is true and $\neg v_i$ is false, we conclude that r' is true. Thus, the disjunction $r \vee r'$ is also true.
  - If v_i is false, then from the fact that $v_i \vee r$ is true and v_i is false, we conclude that r is true. Thus, the disjunction $r \vee r'$ is also true.

  Thus, in both cases, the formula $(r \vee r') \,\&\, R$ is satisfied as well.
- If the formula $(r \vee r') \,\&\, R$ is satisfied, this means that for some combination of variables, both R and $r \vee r'$ are true. Thus, either r is true, or r' is true.
  - In the first case, when r is true, we can take v_i to be false. Then, due to the above fact about $\vee$, r is true hence $v_i \vee r$, and $\neg v_i$ is true hence $\neg v_i \vee r'$ is true.
  - In the second case, when r' is true, we can take v_i to be true. Then, due to the above fact about $\vee$, v_i is true hence $v_i \vee r$ is true, and r' is true hence $\neg v_i \vee r'$ is true.

  Thus, in both cases, the formula $(v_i \vee r) \,\&\, (\neg v_i \vee r') \,\&\, R$ is true as well.

The proposition is proven for all the cases.

Proof of Proposition 3. As we mentioned, computing the range of variance under interval uncertainty is NP-hard, but variance is a DUE:

$$V = \frac{x_1^2 + \ldots + x_i^2 + \ldots + x_n^2}{n} - \left(\frac{x_1 + \ldots + x_i + \ldots + x_n}{n}\right)^2.$$

The proposition is proven.

Proof of Proposition 4. In this case, we need to check whether there are values p_ℓ that satisfy the system of linear equations $\sum\limits_{\ell=1}^{k} A_{i\ell} \cdot p_\ell = 0$ and linear inequalities $\underline{p}_\ell \le p_\ell \le \overline{p}_\ell$ (that describe interval constraints on p_ℓ).

It is known that checking consistency of a given system of linear equations and inequalities is a feasible problem, a particular case of linear programming; see, e.g., [1]. Thus, any feasible algorithm for solving linear programming problem solves our problem as well. The proposition is proven.

Proof of Proposition 5. We have already mentioned that finding the range of a quadratic function $f(p_1, \ldots, p_k)$ under interval uncertainty $p_\ell \in \mathbf{p}_\ell$, is NP-hard. It is also true (see, e.g., [9]) that checking, for a given value v_0, where there exists values $p_\ell \in \mathbf{p}_\ell$ for which $f(p_1, \ldots, p_k) = v_0$ is also NP-hard.

We can reduce this NP-hard problem to our problem by considering a very simple system consisting of a single equation $a_{11} \cdot y_1 = b_1$, with $y_1 = 1$, $b_1 = v_0$, and $a_{11} = f(p_1, \ldots, p_k)$. The tuple $x = (1)$ belongs to the solution set if and only if there exist values p_ℓ for which $f(p_1, \ldots, p_k) = v_0$.

The reduction is proven, so our checking problem is indeed NP-hard.

ACKNOWLEDGMENT

The author is thankful to G. Alefeld, V. Kreinovich, and E. Popova for valuable discussions, and to the anonymous referees for important suggestions.

REFERENCES

[1] C. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Boston, Massachusetts, 2009.

[2] S. Ferson, L. Ginzburg, V. Kreinovich, L. Longpré, and M. Aviles, "Computing Variance for Interval Data is NP-Hard", *ACM SIGACT News*, 2002, Vol. 33, No. 2, pp. 108–118.

[3] S. Ferson, L. Ginzburg, V. Kreinovich, L. Longpré, and M. Aviles, "Exact Bounds on Finite Populations of Interval Data", *Reliable Computing*, 2005, Vol. 11, No. 3, pp. 207–233.

[4] A. A. Gaganov, *Computational complexity of the range of the polynomial in several variables*, Leningrad University, Math. Department, M.S. Thesis, 1981 (in Russian).

[5] A. A. Gaganov, "Computational complexity of the range of the polynomial in several variables", *Cybernetics*, 1985, pp. 418–421.

[6] E. Hansen, "Sharpness in interval computations", *Reliable Computing*, 1997, Vol. 3, pp. 7–29.

[7] 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.

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

[9] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1998.

[10] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*, SIAM Press, Philadelphia, Pennsylviania, 2009.

[11] H. T. Nguyen and E. A. Walker, *First Course on Fuzzy Logic*, CRC Press, Boca Raton, Florida, 2006.

[12] E. D. Popova, "Explicit Characterization of a Class of Parametric Solution Sets", *Compt. Rend. Acad. Bulg. Sci.*, 2009, Vol. 62, No. 10, pp. 1207–1216.

[13] E. D. Popova, "Characterization of paremetric solution sets", *Abstracts of the 14th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics SCAN'2010*, Lyon, France, September 27–30, 2010, pp. 118–119.

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