If an Exact Interval Computation Problem Is NP-Hard, then the Approximate Problem Is Also NP-Hard: A Meta-Result

Aline B. Loreto[1], Laira V. Toscani[1], Leila Robeiro[1],
Dalcídio M. Cláudio[2], Liara S. Leal[2],
Luc Longpré[3], and Vladik Kreinovich[3]

[1]Institute of Computer Science, PPGC
URFGS – Public University of Rio Grande do Sul, Brazil
{loreto,laira,leila}@inf.ufrgs.br

[2]PUCRS – Pontifical University Catholic of the Rio Grande do Sul, Brazil
dalcidio@inf.pucrs.br, liara@pucrs.br

[3]Computer Science, University of Texas, El Paso, TX 79968, USA
longpre@cs.utep.edu, vladik@utep.edu

Abstract

In interval computations, usually, once we prove that a problem of computing the exact range is NP-hard, then it later turns out that the problem of computing this range with a given accuracy is also NP-hard. In this paper, we provide a general explanation for this phenomenon.

Formulation of the problem. One of the main problems of interval computations is, given a function $f(x_1, \ldots, x_n)$ and n intervals $\mathbf{x}_i$, to compute the range $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of possible values of f when $x_i \in \mathbf{x}_i$.

In interval computations, many subclasses of this general problem are NP-hard: e.g., the problem of computing the range of a quadratic function, the problem of computing the range of the solution to the system of linear questions with linear coefficients, the problem of computing the range of variance with interval data, etc.; see, e.g., [1, 2].

Usually, once we prove that a problem of computing the exact range is NP-hard, then it later turns out that the problem of computing this range with a given accuracy is also NP-hard.

In theory of computing in general, it is possible that a problem is NP-hard but its approximation is easy to solve; several packing and scheduling problems have this property; see, e.g., [3]. In this paper, we provide a general explanation why in interval computations, the introduction of approximations does not make the problem much easier.

How NP-hardness is proved in general. In general, most proofs of NP-hardness reduce a known discrete NP-complete problem to the problem in question.

Definition 1. *By a* discrete problem*, we mean the following problem: we are given a discrete object g, and we need to find a discrete object o such that $P(g, o)$ is true, where the property P can be checked in polynomial time.*

For example, we may start with a propositional satisfiability problem in which g is a propositional formula, and o are the values of the propositional ("yes"-"no") variables that make it true.

In most interval computation proofs, this reduction is usually set up in the way that is formally described below. We show that in this case, the approximate interval computation problem is also NP-hard.

Definition 2. *By an* algebraic function of n variables, *we mean an expression $P(x_1, \ldots, x_n, y)$ that is formed from $n+1$ real-valued variables by using arithmetic operations $+$, $-$, $\cdot$, and $/$, and functions* min *and* max, *and for which for every $x = (x_1, \ldots, x_n)$, there exists one and only one y for which $P = 0$ – i.e., for which the equality $P = 0$ determines y as a function of $x_1, \ldots, x_n$.*

Comment. The corresponding function (explicit or implicit) will be denoted by f_P.

We allow implicit functions since we want to also cover the cases when the value y comes, e.g., from solving a system of linear equations. Explicit functions $y = f(x_1, \ldots, x_n)$ are a particular case of this definition – we can, e.g., take $P(x_1, \ldots, x_n) = y - f(x_1, \ldots, x_n)$.

Definition 3. *By an* exact interval computation problem, *we mean a class of tuples $\langle P, \mathbf{x}_1, \ldots, \mathbf{x}_n \rangle$, where P is an algebraic expression of n intervals, and $\mathbf{x}_i$ are rational-valued intervals such that for each of these problems, the values*

$$\underline{r} \stackrel{\text{def}}{=} \inf\{f_P(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\} \text{ and } \overline{r} \stackrel{\text{def}}{=} \sup\{f_P(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$$

are rational numbers.

Definition 4. *A* reduction *from a discrete problem P to an exact computation problem $\mathcal{P}$ means that for each instance of the discrete problem P – i.e., for each object g – we feasibly (i.e., in polynomial time) form an instance P_g of the corresponding interval computation problem, so that from the solution to P_g, we can feasibly check whether the original instance of the discrete problem has a solution.*

Definition 5. *We say that an exact interval computation problem $\mathcal{P}$ has a* traditional NP-hardness proof *if there is a reduction from a discrete NP-complete problem P to this problem which has the following properties:*

- *the original instance of the discrete problem has a solution if and only if the range $[\underline{r}, \overline{r}]$ of the corresponding interval problem satisfies the inequality $\underline{r} \le a(g)$ (or, alternatively, $\overline{r} \ge a(g)$), where $a(g)$ is a feasibly computable rational-valued function of g;*
- *based on a solution o of the discrete problem, we can feasibly compute the values $x_1, \ldots, x_n$ for which $f_g(x_1, \ldots, x_n) \le a(g)$ (corr., $f_g(x_1, \ldots, x_n) \ge a(g)$);*
- *vice versa, if we know the values x_i for which $f_g(x_1, \ldots, x_n) \le a(g)$ (corr., $f_g(x_1, \ldots, x_n) \ge a(g)$), then we can feasibly compute a solution o to the original discrete problem;*
- *the value $\underline{r}$ (coor., $\overline{r}$) is attained at one of the discretely many points $x(d)$, where d is a discrete string of length n, and $x(d)$ is a feasible function of d.*

Comment. In interval computations, most known NP-hardness results have proofs which are traditional in this sense. For example, for a quadratic function $f_g(x_1, \ldots, x_n)$, for each variable x_i, the minimum of f_g is attained if either (1) $x_i = \underline{x}_i$ or (2) $x_i = \overline{x}_i$, or (3) $\partial f / \partial x_i = 0$. If we know, for each i, which of these cases $d_i \in \{1, 2, 3\}$ occurs, then we get a system of linear equations from which we can feasibly find the corresponding point $x(d)$ (where $d \stackrel{\text{def}}{=} d_1 \ldots d_n$).

Definition 6. *By an* accuracy function, *we mean a feasible function ε that maps every natural number n into a positive rational number $\varepsilon(n)$.*

Definition 7. *Let $\mathcal{C}$ be an exact interval computation problem, and let ε be an accuracy function. By the* corresponding approximate interval computations problem, *we mean the following problem: given $\langle P, \mathbf{x}_1, \ldots, \mathbf{x}_n \rangle \in \mathcal{C}$, compute rational numbers $\underline{a}$ and $\overline{a}$ which are $\varepsilon(n)$-close to the values $\underline{r}$ and $\overline{r}$ corresponding to the exact problem.*

Result. *If an exact interval computation problem $\mathcal{P}$ has a traditional NP-hardness proof, then there exists an accuracy function for which the corresponding approximate interval computations problem is also NP-hard.*

Proof. We know that $a(g)$ can be computed in polynomial time, i.e., in time $\le T_1(n)$, where n is the size of the problem and $T_1(n)$ is a polynomial. We also know that $\underline{r} = x(d)$, hence $\underline{r}$ can also be computed in polynomial time ($\le T_2(n)$) – once we know a sequence d.

This means that once we know d, then we can compute both the numerators and the denominators of both fractions $p/q \stackrel{\text{def}}{=} \underline{r}$ and $p'/q' \stackrel{\text{def}}{=} a(g)$ in time $\le T(n) \stackrel{\text{def}}{=} T_1(n) + T_2(n)$. Since in one computation step, we can produce at most one bit of an integer, this means that the values p, q, p', and q' cannot have more than $T(n)$ binary digits – hence $p, q, p', q' \le 2^{T(n)}$.

Let us show that if we can compute $\underline{r}$ with an accuracy $2^{-2T(n)-2}$, i.e., if we can compute a value $\widetilde{r}$ for which $|\widetilde{r} - \underline{r}| \le (1/4) \cdot 2^{-2T(n)}$, then we will still be able to check whether $\underline{r} \ge a(g)$ – and thus, the approximate interval computations problem is still NP-hard.

Indeed, we need to check whether $\underline{r} \le a(g)$ or $\underline{r} > a(g)$. If $\underline{r} \le a(g)$, then $\widetilde{r} \le a(g) + (1/4) \cdot 2^{-2T(n)}$.

Let us now consider the case when $\underline{r} > a(g)$. In this case, the positive difference $\underline{r} - a(g) = p/q - p'/q'$ is equal to $r/(q \cdot q')$ for some positive integer $r > 0$. Since $r > 0$, we have $r \ge 1$, hence $\underline{r} \ge a(g) + 1/(q \cdot q')$. Since $q, q' \le 2^{T(n)}$, we have $1/(q \cdot q') \ge 2^{-2T(n)}$, so we conclude that in this case $\underline{r} \ge a(g) + 2^{-2T(n)}$. Hence, $\widetilde{r} \ge \underline{r} - (1/4) \cdot 2^{-2T(n)} \ge a(g) + (3/4) \cdot 2^{-2T(n)}$.

So, we will have either $\widetilde{r} \le a(g) + (1/4) \cdot 2^{-2T(n)}$ or $\widetilde{r} \ge a(g) + (3/4) \cdot 2^{-2T(n)}$. Based on the rational value $\widetilde{r}$, we can feasibly tell which of the two inequalities is true – and thus, we will be able to tell whether $\underline{r} \ge a(g)$.

Thus, the original NP-hard problem is reduced to the approximate interval computation problem of computing $\widetilde{r}$ – hence, this approximate problem is NP-hard. The statement is proven.

Comment. We have shown that the approximate interval computation problem is NP-hard for some small accuracy ε. A natural question is: is it NP-hard for all accuracies? The answer to this question depends on the problem.

Some interval computation problems are scale-invariant – e.g., the problem of computing the range of a quadratic polynomial. In this case, if we know how to compute the range of an arbitrary instance problem f with an accuracy ε, then, in particular, for every $\delta > 0$, we will be able to compute the range of $f' \stackrel{\text{def}}{=} (f/\varepsilon) \cdot \delta$ with an accuracy ε – which is equivalent to computing the range of f with accuracy δ.

For such problems, if computing the range with any given accuracy is NP-hard, then it is NP-hard for any other accuracy as well.

Other problems are not scale-invariant – e.g., if we restrict ourselves only to quadratic polynomials with coefficients from $[-1, 1]$. In some such cases, we have a prior bound B on the range. In this situation, the problem of computing the range with, e.g., accuracy B is trivial – just return $\widetilde{r} = 0$.

Acknowledgments. This work was supported in part by NASA under cooperative agreement NCC5-209, NSF grant EAR-0225670, NIH grant 3T34GM008048-20S1, and Army Research Lab grant DATM-05-02-C-0046.

References

[1] 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.

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

[3] V. Vazirani, *Approximation algorithms*, Springer-Verlag, Barlin, 2001.