8-1-2009

Diagonalization is Also Practically Useful: A Geometric Idea

Martine Ceberio
University of Texas at El Paso, mceberio@utep.edu

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

Comments:
Technical Report: UTEP-CS-09-26

DIAGONALIZATION IS ALSO PRACTICALLY USEFUL: A GEOMETRIC IDEA

Martine Ceberio and Vladik Kreinovich
Department of Computer Science
University of Texas, El Paso, TX 79968, USA
mceberio@utep.edu, vladik@utep.edu

Cantor's diagonalization idea: reminder. Most mathematicians are familiar with diagonalization from Cantor's proof that the set of all real numbers is not countable. Indeed, if it was countable, i.e., if we could enumerate all (decimal) real numbers $d_1 d_2 \ldots d_i . f_1 \ldots f_j \ldots$ into a sequence $d_1^{(k)} d_2^{(k)} \ldots d_i^{(k)} . f_1^{(k)} \ldots f_j^{(k)} \ldots$, $k = 1, 2, \ldots$, then we would be able to design a new real number which is *not* in this sequence, by:

- first forming a diagonal fraction $0.f_1^{(1)} f_2^{(2)} \ldots f_k^{(k)} \ldots$ and then
- changing each digit to a different one: e.g., by adding 1 to every digit (with the understanding that 9 + 1 turns into 0).

This procedure is called *diagonalization*, because if we form an (infinite) matrix by placing all the digits of the k-th number in the k-th row, the new number depends only on what is in the diagonal of this matrix:

0	.	$\boxed{0}$	0	0	...
0	.	3	$\boxed{3}$	3	...
3	.	1	4	$\boxed{4}$	...
			...		
0	.	$\underline{0}$	$\underline{3}$	$\underline{4}$	...
0	.	1	4	5	...

The resulting new number is different from all the numbers from the original sequence: for each k, it is different from the k-th number because of the difference in the k-th digit. This contradicts to

our assumption that the original sequence covers all possible real numbers.

Diagonalization is normally used only in proofs. Diagonalization is actively used in mathematics and in theoretical computer science. For example, the original proof of the halting problem (that it is not possible to algorithmically decide whether a given Turing machine halts) is based on diagonalization; see, e.g., (Papadimitriou 1994).

In all known applications, diagonalization is used for a purely theoretical purpose: to prove results. In this paper, we show that, contrary to a common perception, diagonalization is also (implicitly) used in practical algorithms. Specifically, diagonalization implicitly appear in the interval-based constraint approach to solving system of equations; see, e.g., (Jaulin *et al.* 2001).

Interval-based constraint approach to solving system of equations: a reminder. Suppose that we have a system of equations with n unknowns. In this approach, first, use a standard compiler algorithm to parse each equation into elementary constraints.

For example, to parse an equation $x - x^2 = 0.5$ for $x \in [0,1]$ (with, by the way, has no solutions), we take into account that in order to compute $x - x^2$, the computer will first compute $r = x^2$, and then compute the difference $x - r$ (which is equal to 0.5). Thus, we get two elementary constraints: $r = x^2$ and $0.5 = x - r$.

Each elementary constraint enables us to describe the value of each of the unknowns in terms of the values of the others.

In the above example, from the constraint $r = x^2$, we extract two rules: (1) $x \to r = x^2$ and (2) $r \to x = \sqrt{r}$; from the constraint $0.5 = x - r$, we extract two more rules: (3) $x \to r = x - 0.5$ and (4) $r \to x = r + 0.5$.

We start with the original ranges $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ for the unknowns x_i. In the above example, we start with $\mathbf{x} = [0,1]$ and $\mathbf{r} = (-\infty, \infty)$. Then, we cycle through all the rules, and use the ranges for the other unknowns to contract the range of the variable described by this rule.

Rule (1), with $x \in [0,1]$, leads to $r \in \mathbf{r} = [0,1]^2 = [0,1]$. We already know that $r \in (-\infty, \infty)$, so we conclude that r belongs

to the intersection $\mathbf{r}_{\text{new}} = (-\infty, \infty) \cap [0,1] = [0,1]$. Then, this procedure leads to the following results:

(2) $\mathbf{x}_{\text{new}} = \sqrt{[0,1]} \cap [0,1] = [0,1]$ – no change.

(3) $\mathbf{r}_{\text{new}} = ([0,1] - 0.5) \cap [0,1] = [-0.5, 0.5] \cap [0,1] = [0, 0.5]$.

(4) $\mathbf{x}_{\text{new}} = ([0, 0.5] + 0.5) \cap [0,1] = [0.5, 1] \cap [0,1] = [0.5, 1]$.

(1) $\mathbf{r}_{\text{new}} = [0.5, 1]^2 \cap [0, 0.5] = [0.25, 0.5]$.

(2) $\mathbf{x}_{\text{new}} = \sqrt{[0.25, 0.5]} \cap [0.5, 1] = [0.5, 0.71]$;
(we round $\underline{a}$ down $\downarrow$ and $\overline{a}$ up $\uparrow$, to guarantee enclosure);

(3) $\mathbf{r}_{\text{new}} = ([0.5, 0.71] - 0.5) \cap [0.25, .5] = [0.0.21] \cap [0.25, .5]$,
i.e., $\mathbf{r}_{\text{new}} = \emptyset$.

Our conclusion is that the original equation has no solutions.

For the equation $x - x^2 = 0$, $x \in [0,1]$, a similar procedure leads to the interval $\mathbf{x} = [0,1]$ which contains both solutions $x = 0$ and $x = 1$. If we bisect $\mathbf{x} = [0,1]$ into $[0, 0.5]$ and $[0.5, 1]$, then for each subinterval, we get the corresponding solution.

Constraint approach: a limitation. Our objective is to find the set S of all possible tuples $x = (x_1, \ldots, x_n)$ that satisfy all the constraints.

Sometimes, the above constraint technique leads us exactly to this set S. However, in general, in the constraint approach, we only restrict each of the unknowns x_i by an appropriate set X_i. In the ideal case, each of these restrictions is "exact" in the sense that each set X_i coincides with the set of all possible values x_i when $x \in S$:

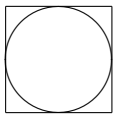
$$X_i = \{x_i : (x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \in S\},$$

i.e., with the i-th projection of the set S: $X_i = \pi_i(S)$.

After applying this approach, the only thing we know about the resulting tuples x is that $x_1 \in X_1$, …, $x_n \in X_n$. In other words, instead of the original set S, we only know that the every solution tuple x belongs to the (larger) set

$$X_1 \times \ldots \times X_n = \pi_1(S) \times \ldots \times \pi_n(S). \qquad \text{(C)}$$

For example, if the set S is closed and connected, then each projection $\pi_i(S)$ is an interval, so the resulting set is the "interval hull" of the set S: the smallest "box" $[\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n]$ containing S:



Relation to diagonalization. We will show that the above formula (C) is directly related to diagonalization. Indeed, diagonalization can be formally defined as follows: we start with n tuples $x^{(k)} = (x_1^{(k)}, \ldots, x_n^{(k)})$, $1 \le k \le n$, and we produce a new tuple

$$\mathrm{diag}(x^{(1)}, \ldots, x^{(n)}) \stackrel{\mathrm{def}}{=} (x_1^{(1)}, \ldots, x_i^{(i)}, \ldots, x_n^{(n)}).$$

For each set S, we can define the range $\mathrm{diag}(S, \ldots, S)$ of this operation in the usual way:

$$\mathrm{diag}(S, \ldots, S) \stackrel{\mathrm{def}}{=} \{\mathrm{diag}(x^{(1)}, \ldots, x^{(n)}) : x^{(1)} \in S, \ldots, x^{(n)} \in S\}.$$

One can easily check that

$$\mathrm{diag}(S, \ldots, S) = \pi_1(S) \times \ldots \times \pi_n(S),$$

i.e., that the hull computed by the constraints techniques is *exactly* the diagonalization result.

Open questions and future work. In some problems, it is beneficial, instead of simply limiting the range of each unknown x_i, to also limit the range of pairs (x_i, x_j); see, e.g., (Ceberio, Ferson, *et al.* 2007) and (Ceberio, Kreinovich, *et al.* 2007). In this case, instead of n sets X_i, we also get sets $X_{ij} = \pi_{ij}(S)$ of all possible values of the pairs (x_i, x_j). Then, instead of the original set S, we have a set

$$\{(x_1, \ldots, x_n) : (x_i, x_i) \in X_{ij} \text{ for all } i, j\}.$$

This set is smaller than the diagonalization result. If this set is still too large, we can consider triples, etc.

It would be great to analyze both the geometric meaning of this new set and its possible relation to diagonalization-type constructions.

Another open question is related to the possibility of an algebraic reformulation of diagonalization. For tuples of length two, the diagonalization operation $*$ satisfies the properties $a * a = a$ and $a * (b * c) = (a * b) * c = a * c$. One can show that in some reasonable sense, these properties uniquely determine a diagonalization process. Indeed, if we have an operation $*$ on a set X satisfying these properties, then we can pick an element $x_0 \in X$ and define $\pi_1(a) \stackrel{\text{def}}{=} a * x_0$ and $\pi_1(b) \stackrel{\text{def}}{=} x_0 * b$. Then, for ever a and b, we have

$$a * b = (a * x_0) * (x_0 * b) = \pi_1(a) * \pi_2(b);$$

so, the value $a*b$ is uniquely determined by the first projection $\pi_1(a)$ of a and by the second projection $\pi_2(b)$ of b.

It is possible to have a similar algebraic representation for a diagonalization operation for triples. For example, for triples, we have the rules

$$*(a,a,a) = a, \;\; *(*(a,b,c),d,e) = *(a,d,e),$$

$$*(a,(b,c,d),e) = *(a,c,e), \;\; *(a,b,*(c,d,e)) = *(a,b,e).$$

One can check that these rules imply that

$$*(a,b,c) = *(\pi_1(a), \pi_2(b), \pi_3(c)),$$

where

$$\pi_1(a) \stackrel{\text{def}}{=} *(a,x_0,x_0), \;\; \pi_2(b) \stackrel{\text{def}}{=} *(x_0,b,x_0), \text{ and}$$

$$\pi_3(c) \stackrel{\text{def}}{=} *(x_0,x_0,c).$$

It is desirable to study the algebraic properties of such operations, as well as their relation to the recombination (crossover) procedure in genetic algorithms, where we, e.g., combine the first part of a gene sequence $x^{(1)}$ with the remaining part borrowed from another

gene sequence $x^{(2)}$ – just like it happens when the biological genes recombine in a child; see, e.g., (Affenzeller *et al.* 2009).

Acknowledgment. This work was supported in part by the National Science Foundation grant HRD-0734825, by Grant 1 T36 GM078000-01 from the National Institutes of Health, and by Grant 5015 from the Science and Technology Centre in Ukraine (STCU), funded by European Union.

References

M. Affenzeller, S. Winkler, S. Wanger, and A. Beham, *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*, CRC Press, Boca Raton, 2009.

M. Ceberio, S. Ferson, V. Kreinovich, S. Chopra, G. Xiang, A. Murguia, and J. Santillan, "How To Take Into Account Dependence Between the Inputs: From Interval Computations to Constraint-Related Set Computations, with Potential Applications to Nuclear Safety, Bio- and Geosciences", *Journal of Uncertain Systems*, 2007, Vol. 1, No. 1, pp. 11–34.

M. Ceberio, V. Kreinovich, A. Pownuk, and B. Bede, "From Interval Computations to Constraint-Related Set Computations: Towards Faster Estimation of Statistics and ODEs under Interval, p-Box, and Fuzzy Uncertainty", In: P. Melin, O. Castillo, L. T. Aguilar, J. Kacprzyk, and W. Pedrycz (eds.), *Foundations of Fuzzy Logic and Soft Computing*, Proceedings of the World Congress of the International Fuzzy Systems Association IFSA'2007, Cancun, Mexico, June 18–21, 2007, Springer Lecture Notes on Artificial Intelligence, 2007, Vol. 4529, pp. 33–42.

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.

C. H. Papadimitriou, *Computational Complexity*, Addison Wesley, Reading, Massachusetts, 1994.