Interval methods for solving quantified nonlinear problems for control engineering and machine learning

Bartłomiej Jacek Kubica

Institute of Information Technology
Warsaw University of Life Sciences – SGGW
Poland

Interval Online Seminar on
Interval Methods in Control Engineering
19th of November 2021

Problem under solution

Find **all** $x \in \mathbb{R}^n$, satisfying the condition $P(x)$, i.e., find the set $\{x \in \mathbb{R}^n | P(x)\}$.

where *P (x)* is a predicate formula with a free variable *x*, i.e., free variables: $x_1, \ldots, x_n$.

It can contain bound variables, also (we shall call them `parameters’).

Example problems

$$\{x \in X \mid h(x) = 0\}$$

$$\{x \in X \mid f(x) \in [\underline{y}, \overline{y}]\}$$

$$\{x \in X \mid (\forall t \in X)\ (f(x) \leqslant f(t))\}$$

$$\{x \in X \mid (\forall t \in X)\ (\forall i = 1, \ldots, N\ \ f_i(x) \leqslant f_i(t)) \vee (\exists i\ \ f_i(x) < f_i(t))\}$$

$$\{x \in X \mid (\forall i = 1, \ldots, n)\ (\forall x'_i \in \boldsymbol{x_i} \subseteq \mathbb{R}^{k_i})\ (f_i(x_{\setminus i}, x'_i) \geqslant f_i(x))\}$$

where:

$$X \subseteq \mathbb{R}^n, \quad h : \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad f, f_1, \ldots f_N : \mathbb{R}^n \rightarrow \mathbb{R}$$

Proposed algorithm

- V. Kreinovich, B. J. Kubica, *From computing sets of optima, Pareto sets and sets of Nash equilibria to general decision-related set computations*, Journal of Universal Computer Science, Vol. 16, pp. 2657 – 2685 (2010).
- B. J. Kubica, *A class of problems that can be solved using interval algorithms*, SCAN 2010, Computing, Vol. 94 (2-4), pp. 271 – 280 (2012).
- B. J. Kubica, *Interval Methods for Solving Nonlinear Constraint Satisfaction, Optimization and Similar Problems*, monograph, ISBN 978-3-030-13795-3, Springer, 2019.
- B. J. Kubica, *Interval methods for solving various kinds of quantified nonlinear problems*, in: *Beyond Traditional Probabilistic Data Processing Techniques: Interval, Fuzzy etc. Methods and Their Applications*, Springer, 2020.

Proposed algorithm

- We shall use the name generalized branch-and-bound method or branch-and-bound type method.
- Several algorithms, described in the literature, are its specific instances:
  - Branch-and-bound method.
  - Branch-and-prune method.
  - \`Nested’ b&b or b&p (for parameters).
  - SIVIA – Set Inversion Via Interval Analysis (Jaulin, 1993).
  - PPS – Partitioning Parameter Set (дроблене параметров; Калмыков, 1982).

Problem

- It is easy to understand that interval methods can be used to verify inequalities and their systems.
- Using proper theorems, we can also verify equations and their systems (interval Newton, Kantorovich, Miranda, Borsuk…).
- We can extend it to problems of the form:

$$\{x \in X \mid (\forall t \in [t_0, t_f]) \ (f(x,t) \leqslant 0)\}$$

- But how can it be used to solve quantified problems, like global optimization:

$$\{x \in X \mid (\forall t \in X) \ (f(x) \leqslant f(t))\}?$$

Problem

- What are you talking about? Interval methods have been used for GO since forever!

Problem

- What are you talking about? Interval methods have been used for GO since forever!
- Actually, not quite.
- The problem we solve in these algorithms is **not**:

$$\left\{ x \in X \mid \left(\forall t \in X \right) \ \left(f\left(x \right) \leqslant f\left(t \right) \right) \right\}$$

Problem

- What are you talking about? Interval methods have been used for GO since forever!
- Actually, not quite.
- The problem we solve in these algorithms is **not**:

  $$\{x \in X \mid (\forall t \in X)\ (f(x) \leqslant f(t))\}$$

- No, really!
- It is, actually:

  $$\{x \in X \mid\ (f(x) \leqslant y_o)\},$$

  where y_0 is a parameter, that we need to estimate first: $y_0 = f(x_0)\ \mathit{for\ some}\ x_0 \in X\ \mathit{and}\ \neg(\exists x'\, f(x') < f(x_0) - \epsilon)$

What is the relationship between these two problems?

- They are not equivalent!
- $\{x \in X \mid (f(x) \leqslant y_o)\}$ is quantifier-free.
- $\{x \in X \mid (f(x) \leqslant y_o)\}$ is implied by $\{x \in X \mid (\forall t \in X)\ (f(x) \leqslant f(t))\}$
- $\{x \in X \mid (f(x) \leqslant y_o)\}$ is weaker.

What is the relationship between these two problems?

- They are not equivalent!
- $\{x \in X \mid \ (f(x) \leqslant y_o)\}$ is quantifier-free.
- $\{x \in X \mid \ (f(x) \leqslant y_o)\}$ is implied by $\{x \in X \mid (\forall t \in X) \ (f(x) \leqslant f(t))\}$
- $\{x \in X \mid \ (f(x) \leqslant y_o)\}$ is weaker.
- $\{x \in X \mid \ (f(x) \leqslant y_o)\}$ is a result of **approximate quantifier elimination**.

What is the relationship between these two problems?

- They are not equivalent!
- $\{x \in X \mid \; (f(x) \leqslant y_o)\}$ is quantifier-free.
- $\{x \in X \mid \; (f(x) \leqslant y_o)\}$ is implied by $\{x \in X \mid (\forall t \in X) \; (f(x) \leqslant f(t))\}$
- $\{x \in X \mid \; (f(x) \leqslant y_o)\}$ is weaker.
- $\{x \in X \mid \; (f(x) \leqslant y_o)\}$ is a result of **approximate quantifier elimination**.
- It is the Herbrand form of the original problem: $\{x \in X \mid (\forall t \in X) \; (f(x) \leqslant f(t))\}$

Herbrand form

- Conjunction (for $P(x) \equiv (\forall t)\, R(x,t)$) or dyzjunction (for $P(x) \equiv (\exists t)\, R(x,t)$).
- The Herbrand form of a formula is commonly used in logic, to prove that this formula is a tautology.
  - A formuła is a tautology when its Herbrand form is a tautology.
- But it can also be used for the approximation of a formula:

$$(\forall t)\ R(x,t) \Rightarrow R(x,t_1) \wedge R(x,t_2) \wedge \ldots \wedge R(x,t_N),$$
$$(\exists t)\ R(x,t) \Leftarrow R(x,t_1) \vee R(x,t_2) \vee \ldots \vee R(x,t_N).$$

Herbrand form

- Most of the aforementioned problems had a universal quantifier:

$$P(x) \equiv (\forall t)\, R(x,t)$$

- Later in the talk, we shall meet a problem with existential quantifier, as well:

$$P(x) \equiv (\exists t)\, R(x,t)$$

- Please mind, the relationships are different:

$$(\forall t)\; R(x,t) \;\Rightarrow\; R(x,t_1) \wedge R(x,t_2) \wedge \ldots \wedge R(x,t_N),$$
$$(\exists t)\; R(x,t) \;\Leftarrow\; R(x,t_1) \vee R(x,t_2) \vee \ldots \vee R(x,t_N).$$

Comments

- Obviously, we have an analogous situation for other aforementioned problems: when approximating Pareto sets of a multicriteria problem, or seeking game solutions, we also use Herbrand expansions.
  - The Herbrand formuła for these problems is more complex than for the simple, unicriterion global optimization.
  - We have to determine values of a higher number of parameters (also called `shared quantities’ in my other publications).

Generic algorithm

Lpos = {}; *Lverif* = {}; *Lcheck* = {};
// Phase I
while (there are boxes to consider) **do**
    pop (***x***);
    process (***x***); // using interval tools
    if (***x*** was verified to contain a solution/a point satisfying some necessary conditions) **then** push (*Lverif*, ***x***);
    else if (***x*** is verified not to contain solutions) **then**
        if (***x*** may be necessary in phase II) **then** push (*Lcheck*, ***x***);
        else discard ***x*;**
    end if
    if (***x*** was discarded or stored) **then** pop (***x***)**;**
    else if (diam (***x***) < **ε) then** push (*Lpos*, ***x***);
    else
        bisect (***x***, ***x1***, ***x2***); push (***x2***); ***x*** = ***x1***;
    end if
end while
// Phase II – verification of *P(x)* for stored solution candidates
for each (***x*** in *Lverif*   *Lpos*) **do**
    if (***x*** does not contain a solution) **then** discard ***x***;
end for each

What does the algorithm result in?

- Two lists:
  - `Lverif` – the list of boxes verified (certified) to contain a solution,
  - `Lpos` – the list of boxes possibly containing a solution.
- What conditions have to be verified so that the solution was `verified’?
- There can be more than two lists, in general:
  - Some conditions are verified, some are not.
  - We classify points of the domain into more than two classes.

Algorithm's other details

- In what order do we process boxes? Does the order matter?
- How do we store boxes?
- What tools do we use to process a box?
- What information is needed to process a box in phase I?
- What information is needed to verify a box in phase II?
- In particular, what boxes do we store in *Lcheck* – if any?

Algorithm's other details

- All depends on the problem under consideration.
- Does the presence of solution in one area have influence on its presence elsewhere?
  - For equations systems – not really.
  - For optimization problems – it does (the optimum occurs to be local only, if we have found a better point elsewhere).
  - For seeking Pareto sets – also.
  - …
- Obviously, rejection/reduction tests rely on the problem under consideration, also.

Tools to process a single box

- Order of function approximation:
  - 0^{th} order tools – comparing function values.
  - 1^{st} order tools – use of gradients.
  - 2^{nd} order tools – use of Hesse matrices.
  - Higher order tools?
- Operations:
  - Simply, comparing function values.
  - Several versions of the interval Newton operator (*componentwise*, GS) – on various levels.
  - Various constraint satisfaction methods (consistency enforcing, SIVIA, etc.).
  - Tests based on algebraic topology.
  - …

How to make the branch-and-bound-type method efficient?

How to make the branch-and-bound-type method efficient?

- There is a great deal of interval tools.
- All of them give guaranteed (verified) results.
- None of them are intelligent *per se*!

How to make the branch-and-bound-type method efficient?

- There is a great deal of interval tools.
- All of them give guaranteed (verified) results.
- None of them are intelligent *per se*!
- It is crucial to develop an adequate **heuristic** to:
  - choose the interval tools adequate for a specific box,
  - arrange them,
  - parameterize them.

How to make the branch-and-bound-type method efficient?

- The author devoted several papers to design heuristics for two problems:
  - Nonlinear equations systems – especially seeking all solutions of underdetermined systems.
  - Seeking Pareto sets of a multicrietia problem.
- Many tools & versions; several papers.
- The topic is often misunderstood…

Issues in designing heuristics

- Seemingly similar problems might require quite different heuristics.
- The Euclid space of higher dimension has a significantly different geometry than $\mathbb{R}$ or $\mathbb{R}^2$.
  - The interior is small – most results are located near boundaries
  - Even inside a box with small diameter, the distances can be relatively large and the function can change significantly.
  - Bisection hardly reduces distances in the box.

Bisection

- Often, it is assumed that bisection should minimize the diameter of the objective function on resulting boxes.

Bisection

- Often, it is assumed that bisection should minimize the diameter of the objective function on resulting boxes.
- An example of such heuristic is MaxSmear (Shary, 1992; Ratz, 1992; Ratz & Csendes, 1995).
  - Works very well for optimization problems.
  - Works reasonably well for well-determined equations systems.
  - Fails miserably for underdetermined systems.

Bisection

- Often, it is assumed that bisection should minimize the diameter of the objective function on resulting boxes.
- An example of such heuristic is MaxSmear (Shary, 1992; Ratz, 1992; Ratz & Csendes, 1995).
  - Works very well for optimization problems.
  - Works reasonably well for well-determined equations systems.
  - Fails miserably for underdetermined systems.
- In my opinion, the objective of bisection should be defined in a different way: give boxes that are **easy to process** by the used **interval tools**.

Bisection

- For equations solving, the main tool is some kind of the interval Newton operator.
- So, for a single equation in two variables, it might seem reasonable to choose the minimal smear.
- But the convergence…
- A proper policy should take into account several criteria.
- For several, advanced tools, such a policy cannot be too simple…

Bisection

- For example, the heuristic of Kubica, 2012:

find index *j_max* and diameter *w_max* of the longest component;

find index *j_min* and diameter *w_min* of the shortest component;

find index *j_max_nonred* and diameter *w_max_nonred* of the longest component **not reduced** by the latest use of the Newton;

if ((Newton operator reduced no component) or (*w_max* > 1.5 * *w_max_nonred*)) **then return** *j_max*;

else if (*w_max_nonred* > 8 * *w_min*) **then return** *j_max_nonred*;

find index *j* and diameter *w* of the component with the smallest maximal absolute value in all rows of the Jacobi matrix;

if (*w* > 0.1) **then return** *j*;

else return *j_max_noned*;

Bisection

- For Pareto sets seeking, the proper heuristic is quite different:

find the index *i* of the criterion with maximal distance from the set in the criteria space;
find the index *j* and diameter *w* of the component with maximal smear with respect to criterion *i*;
find the index *j_max* and diameter *w_max* of the component with maximal diameter;
if (*w_max* < 8 * *w*) **then return** *j*;
else return *j_max*;

- Reasons: different interval tools, used in the algorithm.

Problems encountered in ML

- The problem of training a classification/regression tools is often formulated as an optimization problem: $\min\limits_{p} \left\| \mathrm{f}\left(\boldsymbol{x}_k\right) - \boldsymbol{y}_k \right\|$

$$\min_{p} \left(\sum_{k=1}^{m} \left(\mathrm{f}\left(\boldsymbol{x}_k\right) - \boldsymbol{y}_k \right)^2 \right) \qquad \text{(LSQ)}$$

$$\min_{p} \left(-\sum_{k=1}^{m} \mathrm{f}\left(\boldsymbol{x}_k\right) \log\left(\boldsymbol{y}_k\right) + \left(1 - \mathrm{f}\left(\boldsymbol{x}_k\right)\right) \log\left(1 - \boldsymbol{y}_k\right) \right) \qquad \text{(KL div)}$$

- Alternatively, a CSP can be used:

$$\mathrm{f}\left(\boldsymbol{x}_k\right) \subseteq \boldsymbol{y}_k, \quad k = 1, \ldots, m$$

Problems encountered in ML

- Yet another possibility: find points satisfying as many constraints of the CSP, as possible.
  - Adequate in the presence of outliers.
- Another problem: find all points satisfying a fuzzy predicate.
  - More lists are needed there: for several alfa-cuts of the fuzzy solution set.
  - Other than that, the geberalized branch-and-bound algorithm can be adopted with minor changes only.

Problems encountered in ML

- A problem with existential quantifier: find points of a dynamical system, where it has periods (of an arbitrary length).
  - In particular, we can consider it for a recurrent neural network (Hopfield, LSTM, Boltzmann machine…).
  - Considering a dynamical system of the form: $x_{k+1} = f(x_k)$, we can formulate the problem, for instance, as follows:

$$\{x \in X \mid (\exists x_1, \ldots, x_n \in X) \wedge (x_1 = f(x)) \wedge (x_2 = f(x_1)) \wedge \cdots \wedge (x = f(x_n))\}$$

  - We consider here cycles of length at most n.
  - How about longer ones?
  - A proper data structure: graph showing possible transitions of values.

Comments and summary

- Interval methods provide us with a toolset for solving a large variety of problems, difficult to formulate or handle using other methods.
- The author is not sure about the general requirements for the formal system, for which we can use interval methods, but it is a large class of systems (the notion of a H-continuous function might be related).
- We can adopt the generalized branch-and-bound algorithm for very sophisticated and versatile problems.
  - This process can hardly be automated, as we need proper heuristics that can be designed by a humen only (only?).

Comments and summary

- This has some impact on our understanding of what computers can solve, and what human are needed to solve.
- This is fascinating, and it (probably) has some philosophical consequences.
- Nevertheless, whenever possible to reduce the problem to either optimization or CSP, it had better be done – for efficiency reasons.
- Oh, and it is worth noting that branch-and-bound type algorithms parallelize well and can utilize current hardware architectures efficiently.