Intervals in constraint programming: some trends and open issues

Djamila Sam-Haroud

Thanks to Gilles Trombettoni for material and comments

Swiss Federal institute of technology, School of core computing, EPFL

June 2009
Outline

1 Interval constraint programming

2 Constraint propagation
   - HC4-revise
   - Box-revise
   - Limitations

3 Trends and open issues
   - The dependency problem
   - The nature of constraints matters
   - Intervals are great . . . but
   - Other sources of improvement

4 Recent solvers
What is interval constraint programming?

A computer science discipline mainly concerned by the design of solvers for **systems equations/inequations over the reals**
☞ with a particular focus on the **completeness** and the **rigor** of computations.
What is interval constraint programming?

- Approach: combining **systematic search** with **contraction/filtering** algorithms
  - the variables’ domains are (generally) represented by **intervals**
  - the contraction/filtering algorithms prune from the intervals the values that are **inconsistent with the constraints**

- The contraction/filtering procedures use techniques from **interval arithmetic** and **interval analysis**.

- Output generated: set of boxes (inner and outer box-covering of the solution space)
Typical solving process: contraction + branching

1. $L \leftarrow \{[b_0]\}$  /* The search starts with an initial box $[b_0]$ */

2. While $L$ is not empty, do:
   1. Choose the first box $[b]$ from $L$.
   2. **Contract/filter** $[b]$ using contraction methods from *interval constraint programming*.
   3. If $[b] \neq \emptyset$ and $diameter([b]) < \epsilon$, then $[b]$ may contain a “solution” (*)
   4. **Branch**: If $[b] \neq \emptyset$ and $diameter([b]) > \epsilon$, then $[b]$ is **split/bisect**, on one dimension (variable), into two sub-boxes $[b_1]$ et $[b_2]$.
   5. $L \leftarrow L \cup \{[b_1]\} \cup \{[b_2]\}$

(*): Interval analysis methods (Interval Newton) are generally used here to certify the existence of a solution in a box.
Example
Example
Example
Example
Example
Example
Example
Example
Outline

1. Interval constraint programming

2. Constraint propagation
   - HC4-revise
   - Box-revise
   - Limitations

3. Trends and open issues
   - The dependency problem
   - The nature of constraints matters
   - Intervals are great . . . but
   - Other sources of improvement

4. Recent solvers
Contraction using constraint propagation

Contraction/filtering + propagation

The most widely used contraction/filtering tools handle one constraint at a time to infer the values to be filtered out from the domains.

☞ A box is contracted with respect to each individual constraint.

When several constraints are involved, the intersection between their boxes is obtained by an incremental constraint propagation algorithm (AC3 like algorithms).

(Typical) contraction procedures (Revise):

- HC4-Revise;
- Box-Revise (BoxNarrow);
Outline

1. Interval constraint programming

2. Constraint propagation
   - HC4-revise
   - Box-revise
   - Limitations

3. Trends and open issues
   - The dependency problem
   - The nature of constraints matters
   - Intervals are great . . . but
   - Other sources of improvement

4. Recent solvers
The **HC4-Revise algorithm**

**HC4-Revise on a constraint** $c$:

- **Forward evaluation**: inferring the interval enclosure of a node on the basis of its children domains.
- **Backward propagation**: refining the interval enclosures of a node on the basis of its parents domains.

**Example**: $((y + (x + z))^2 + 3(x + z)) = 30$
Interval constraint programming

Constraint propagation
- HC4-revise
- Box-revise
- Limitations

Trends and open issues
- The dependency problem
- The nature of constraints matters
- Intervals are great ... but
- Other sources of improvement

Recent solvers
The Box-Revise procedure

Principle:

- Consider the univariate interval constraint 
  \[ c : [f](x, [y_1], ..., [y_k]) = 0 : \] where every variable, except for \( x \), is replaced by their current domains (intervals).
- Reduce \([x]\) by computing the smallest \((l)\) and the largest \((r)\) zero of \( c \).
The Box-Revise algorithm

- Split \([x]\) into sub-intervals \([x_i]\) of width \(\epsilon\) (processed from left to right in the basic version).

- If \(0 \notin [f](\[x_i], [y_1], ..., [y_k])\), then \([x_i]\) is eliminated and the next sub-interval considered.

- If univariate Interval Newton (interval analysis) certifies a unique solution, a zero is identified (at the bounds).
Comparison of Revise procedures

When the box computed by a revise procedure is optimal (as tight as possible with respect to one constraint), it satisfies the $2B/Hull$-consistency property. This property is hard to achieve when a constraint contains multiple occurrences of the same variable, the box is overestimated.

In practice:

- HC4-Revise may not compute the tightest box even if the constraint contains no multiple occurrence of the same variable.

- Box-Revise is more costly but better than HC4-Revise when only one variable occurs several times in the constraint.
Outline

1 Interval constraint programming

2 Constraint propagation
   - HC4-revise
   - Box-revise
   - Limitations

3 Trends and open issues
   - The dependency problem
   - The nature of constraints matters
   - Intervals are great . . . but
   - Other sources of improvement

4 Recent solvers
Main limitations

- Contraction of **individual constraint**
  - The common subexpressions, present in individual constraints or shared by several constraints, are roughly taken into account.

- All types of **constraints** are handled **the same way**
  - Specific features which may help in designing better contraction procedures are under-exploited.

- The output, as set of boxes, may be **too verbose**.
Outline

1. Interval constraint programming

2. Constraint propagation
   - HC4-revise
   - Box-revise
   - Limitations

3. Trends and open issues
   - The dependency problem
   - The nature of constraints matters
   - Intervals are great . . . but
   - Other sources of improvement

4. Recent solvers
Outline

1. Interval constraint programming
2. Constraint propagation
   - HC4-revise
   - Box-revise
   - Limitations
3. Trends and open issues
   - The dependency problem
   - The nature of constraints matters
   - Intervals are great . . . but
   - Other sources of improvement
4. Recent solvers
The dependency problem

Taking into account the way the constraints and their terms relate to each other reduces overestimation.

Examples:

- "Exploiting common subexpressions in Numerical CSPs" (Araya et al., Proc of CP'08)
  - Common sub-expressions are replaced by auxiliary variables
  - The replacement generates useful redundant constraints

- "Interval propagation and search on directed acyclic graphs for numerical constraint solving" (Vu et al., Journal of global opt., 2008)
  - forward-backward evaluation processes are performed on the whole set of constraints rather than on individual constraints
Exploiting common sub-expressions

\[
\begin{align*}
  x^2 + y + (y + x^2 + y^3 - 1)^3 + x^3 &= 2 \\
  \frac{(y^3 + x^2) \times (x^2 + \cos(y)) + 14}{x^2 + \cos(y)} &= 8
\end{align*}
\]

Is (polynomially) rewritten as:

\[
\begin{align*}
  v_2 + v_5^3 + x^3 - 2 &= 0 \\
  \frac{v_3 \times v_4 + 14}{v_4} - 8 &= 0
\end{align*}
\]

\[
\begin{align*}
  v_1 &= x^2 \\
  v_2 &= y + v_1 \\
  v_3 &= v_1 + y^3 \\
  v_4 &= v_1 + \cos(y) \\
  v_5 &= v_2 + y^3 - 1 \\
  v_5 &= -1 + y + v_3
\end{align*}
\]

Redundant constraints, useful for contraction, are generated by replacing common sub-expressions with auxiliary variables.
Interval constraint programming
Constraint propagation
Trends and open issues
Recent solvers

The dependency problem
The nature of constraints matters
Intervals are great . . . but
Other sources of improvement

Propagation on DAGs

Trees:

\{ \sqrt{x} + 2\sqrt{xy} + 2\sqrt{y} \leq 7, \ 0 \leq x^2\sqrt{y} - 2xy + 3\sqrt{y} \leq 2, \ x \in [1, 16], y \in [1, 16] \}.
Propagation on DAGs

are replaced by DAGs:

\[\tau_{N_1} \quad [1, 16] \quad \tau_{N_6} \quad \tau_{N_9} \quad \tau_{N_{10}}\]

\[N_3 \quad N_4 \quad N_5 \quad N_6 \quad N_7 \quad N_8 \quad N_9 \quad N_{10}\]

\[\text{SQRT} \quad \text{SQRT} \quad \star \quad \star \quad \star \quad \star \quad \star \quad \star\]

\[\infty, +\infty \quad \infty, +\infty \quad \infty, +\infty \quad \infty, +\infty \quad \infty, +\infty \quad \infty, +\infty \quad \infty, +\infty \quad \infty, +\infty\]

\[\text{Value} \quad \text{Value} \quad \text{Value} \quad \text{Value} \quad \text{Value} \quad \text{Value} \quad \text{Value} \quad \text{Value}\]

\[\tau_{N_1} \quad [1, 16] \quad \tau_{N_6} \quad \tau_{N_9} \quad \tau_{N_{10}}\]
Using propagation on DAGs (FBPD algorithm):

- potentially *reduces the amount of computation on common subexpressions* shared by constraints,

- and, *explicitly relates constraints to constraints* in the natural way they are composed
Correlating constraints and their terms (potential)

For both approaches:

- generating redundant constraints by replacing common-subexpression with auxiliary variables,

- or propagating on directed acyclic graphs,

☞ significant improvements were obtained compared to the solving techniques based on HC4 and Box.
Outline

1. Interval constraint programming

2. Constraint propagation
   - HC4-revise
   - Box-revise
   - Limitations

3. Trends and open issues
   - The dependency problem
   - The nature of constraints matters
     - Intervals are great . . . but
     - Other sources of improvement

4. Recent solvers
The “nature” of constraints matters

Dedicated algorithms can be used to generate tight bounds for specific constraints (or specific constraints properties)

Examples:

- “Efficient and Safe Global Constraints for Handling Numerical Constraint Systems” (Lebbah et al., SIAM journal or numerical analysis)
  - quadratic and bilinear terms are linearized. The resulting sub-system is contracted by LP techniques

- Generalized to polynomials.

- “Monotonic Hull-consistency” (Araya et al., SWIM’09)
  - uses monotonicity to compute sharper bounds
Outline

1. Interval constraint programming
2. Constraint propagation
   - HC4-revise
   - Box-revise
   - Limitations
3. Trends and open issues
   - The dependency problem
   - The nature of constraints matters
   - Intervals are great . . . but
   - Other sources of improvement
4. Recent solvers
Intervals are great . . . but

Alternative representations of the constraints exists that can contribute to contraction more efficiently:

- parallelepipeds,
- zonotopes,
- linear relaxations,
- convex polyhedral enclosures,
- affine arithmetic forms,
- Bernstein polynomials,
- . . .

Moreover, several different representations can be used cooperatively during a single solving process.

☞ Key issue: Decoupling contraction from constraint representation
Decoupling contraction from constraint representation

Makes it possible to use cooperatively several constraints representation. The output may take more accurate forms. Examples:

- “Contractor Programming” (Chabert and Jaulin, AI journal, 2008)
  - proposes a neat paradigm to decouple contraction from constraint representation

- “Enhancing numerical constraint propagation using multiple inclusion representations” (Vu et al., Annals of mathematics and AI, 2009)
  - generalizes the concepts related to interval forms to provide a common view of different kind of constraints representation
  - proposes a generic combination schema to use several representations cooperatively
Different inclusion representations (interval arithmetic, affine arithmetic ...) can be used to infer redundant constraint sets (PCS: pruning constraint system) to improve contraction:

\[
\begin{align*}
\text{PCS}(N_7, \{I\}) &= \{4\vartheta_{N_1} + 3\vartheta_{N_4} + 2\vartheta_{N_5} = \vartheta_{N_7}; \\
&\quad \vartheta_{N_1} \in [1, 3]; \, \vartheta_{N_4} \in [1, 27]; \\
&\quad \vartheta_{N_5} \in [1, 3]; \, \vartheta_{N_7} \in [9, 9]; \}
\end{align*}
\]

\[
\begin{align*}
\text{PCS}(N_7, \{\mathbb{A}\}) &= \{4\vartheta_{N_1} + 3\vartheta_{N_4} + 2\vartheta_{N_5} = \vartheta_{N_7}; \\
&\quad 2 + \epsilon_1 = \vartheta_{N_1}; \\
&\quad 10 + 5\epsilon_1 + 8\epsilon_2 + 4\epsilon_4 = \vartheta_{N_4}; \\
&\quad 2.125 + \epsilon_2 + 0.125\epsilon_5 = \vartheta_{N_5}; \\
&\quad 42.25 + 19\epsilon_1 + 26\epsilon_2 + 12\epsilon_4 + 0.25\epsilon_5 = \vartheta_{N_7}; \\
&\quad \vartheta_{N_1} \in [1, 3]; \, \vartheta_{N_4} \in [1, 27]; \\
&\quad \vartheta_{N_5} \in [1, 3]; \, \vartheta_{N_7} \in [9, 9]; \\
&\quad \epsilon_i \in [-1, 1] \ (i = 1, 2, 4, 5); \}
\end{align*}
\]
Cooperative use of several representation

Example: combining interval arithmetic, affine arithmetic, and safe linear programming enhance contraction (CIRD[ai] algorithm)

showed significant improvements on several test suites
Outline

1. Interval constraint programming
2. Constraint propagation
   - HC4-revise
   - Box-revise
   - Limitations
3. Trends and open issues
   - The dependency problem
   - The nature of constraints matters
   - Intervals are great . . . but
   - Other sources of improvement
4. Recent solvers
Many research directions

- Further developing the algorithms for computing higher degrees of consistency (like 3B consistency).
- Characterization of new global constraints useful for contraction.
- Developing smart splitting and search strategies.
- Integration of algebraic approach and continuation/homotopy methods.
- Developing smart strategies for the cooperative use of different contraction tools.
- Using clustering techniques to reduce the verbosity of the output, or to restructure the output with respect to particular queries (for example connectedness).
- ...
Outline

1. Interval constraint programming
2. Constraint propagation
   - HC4-revise
   - Box-revise
   - Limitations
3. Trends and open issues
   - The dependency problem
   - The nature of constraints matters
   - Intervals are great . . . but
   - Other sources of improvement
4. Recent solvers
Interval constraint programming solvers

- RealPaver (Benhamou and Granvilliers)
- Gecode (Schulte et al.)
- ICOS (Lebbah)
- Ibex/Quimper (Chabert)
- ...