Enhancing numerical constraint propagation using multiple inclusion representations

Xuan-Ha VU
Djamila Sam-Haroud
Boi Faltings

Artificial Intelligence Laboratory (LIA)
Swiss Federal Institute of Technology in Lausanne (EPFL)

– June, 19th 2008 –

Outline

- **Problem statement←**
- Numerical constraint propagation on DAGs using a single inclusion representation
- Using multiple inclusions on DAGs
  - The CIRD algorithm
- Some experiments
- Conclusions

Problem Formulation

- A numerical constraint satisfaction problem (NCSP): $\mathcal{N} \equiv (\mathrm{V}, \mathrm{D}, \mathrm{C})$
  - $V = (x_1,\ldots, x_n)$: a sequence of **variables**
  - $\mathrm{D} = (D_1,\ldots, D_n)$: a sequence of **domains** of respective variables
    - discrete: $\Delta = (\{1,\ldots, 10\}, \{1,\ldots, 10\})$
    - continuous: $\Delta = ([1, 10], [1, 10])$
  - $C = \{C_1,\ldots, C_m\}$: a set of **constraints**, each is a relation on a subsequence of variables
    - by enumeration: $\mathrm{X} = \{\{(1, 2), (2, 1)\}, \{(1, 2), (2, 4), \ldots, (5, 10)\}\}$
    - by expressions or rules: $\mathrm{X} = \{x + y = 3, 2x - y = 0\}$
- A **solution** of $\mathcal{N}$: a tuple $(a_1,\ldots, a_n) \in D_1 \times\ldots\times D_n$ such that $(a_1,\ldots, a_n) \in C_i$ for all $i = 1, \ldots, m$
  - $(x, y) = (1, 2)$

Solution Methods

- A **complete** method: can find every solution (w.r.t. a reasonable tolerance)
- A **rigorous** method: a complete method dealing with *rounding errors*.
- **Work at LIA:** rigorous methods to compute the solution sets of numerical CSPs (**NCSP**s) of the form

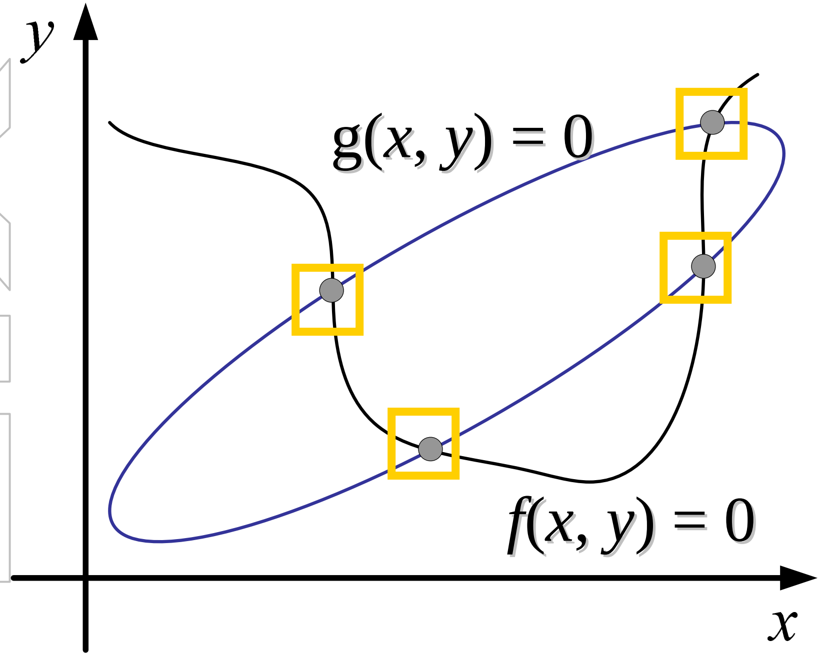
$$\begin{cases} x^2 - 2xy + \sqrt{y} = 0 & \longrightarrow \text{equality} \\ 4x + 3xy + 2\sqrt{y} \le 9 & \longrightarrow \text{inequality} \\ 1 \le x \le 3 \\ y \in [1, 9] \end{cases}$$

continuous variables

continuous domains (connected sets)

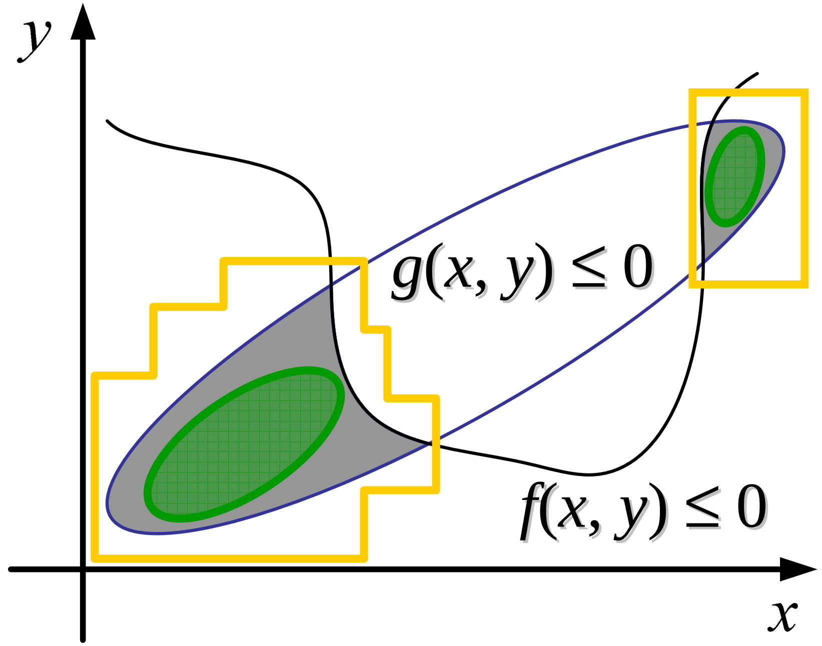
Solution Representation

Outer Approximation $S \subseteq S^{+}$



isolated solutions

Inner Approximation $S^{-} \subseteq S$



continuum of solutions

Branch-and-Prune

The Branch-and-Prune (**BnP**) framework

algorithm BnP(Π)
    Prune(Π);
    ($\Pi_1, \ldots, \Pi_k$) := **Split**(Π);
    for $i = 1$ **to** k **do**
        BnP(Π_i);
    end
end

Branch

Constraint Propagation

Problem reduction,
domain reduction.

Transform a CSP
into equivalent CSPs.

Search

Inclusion representation

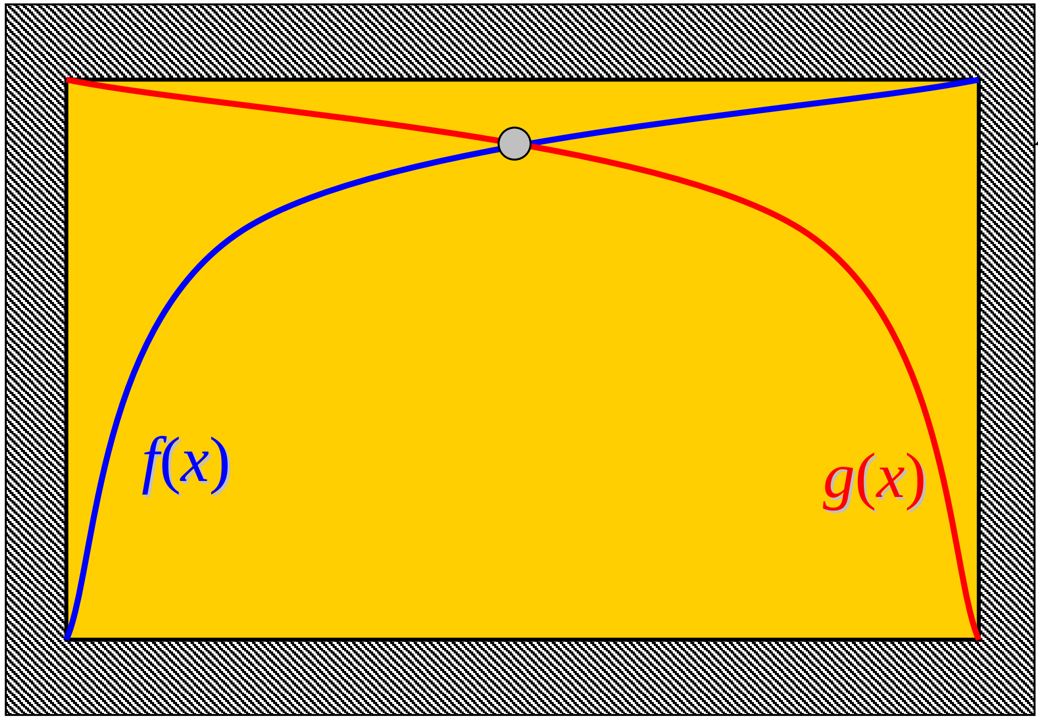
- Conservative enclosure of the solution set of a constraints system
- Can be built using:
  - Interval arithmetic
  - Affine arithmetic (standard, Kolev, Messine, …)
  - Linear relaxations
  - etc..

⇒ used for pruning

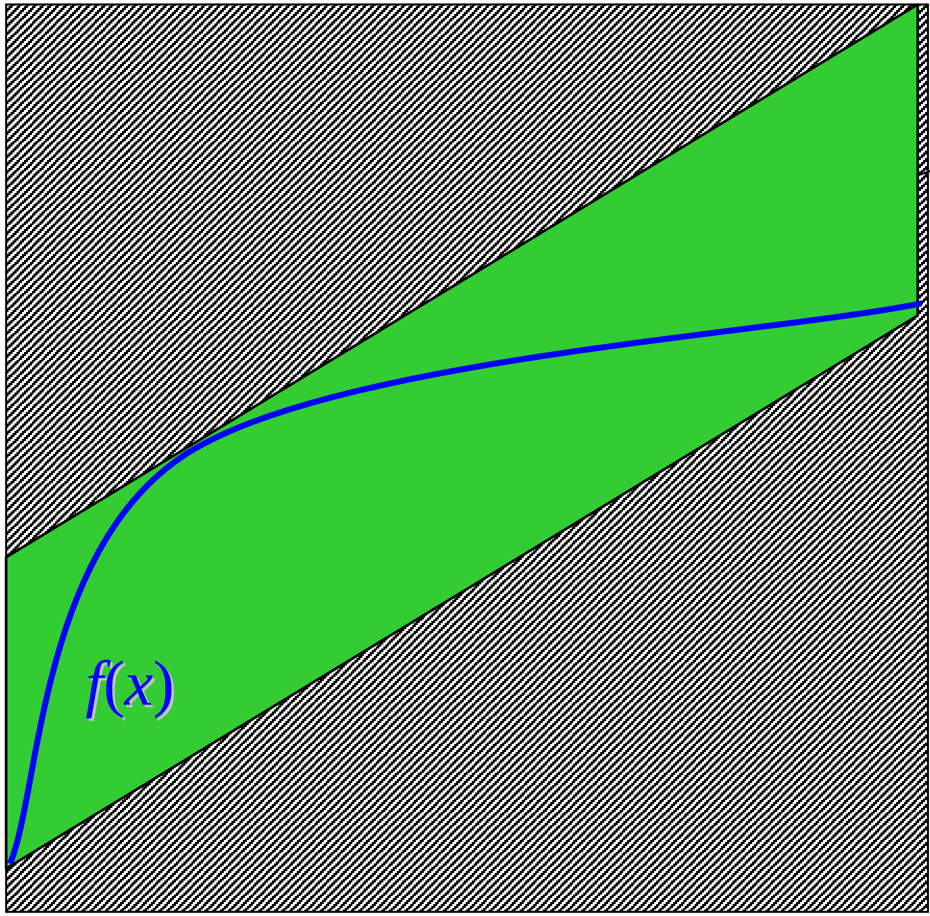
Example (1/2)

interval arithmetic

Moore *et al.*, 1959



Example (2/2)



affine arithmetic

Stolfi *et al.*, 1993

Interval Arithmetic

- A closed interval $\mathbf{x} = [a, b] : x \in \mathbf{x} \Leftrightarrow a \leq x \leq b$.
  - **Interval arithmetic** is an arithmetic that is defined on the set of intervals rather than real numbers.
- Interval arithmetic's operations:
  - Allow to compute elementary operations based on the bounds of intervals, e.g., $\mathbf{x} = [a, b]$, $\mathbf{y} = [c, d] \Rightarrow \mathbf{x} + \mathbf{y} = [a + c, b + d]$.
- The **inclusion property**: $f(\mathbf{x}) \subseteq \mathbf{f}(\mathbf{x})$
  - The range of a real function is included in the value of its interval form.
- **Rounded** interval arithmetic: use outward rounding controls
  - Allow **rigorous enclosures** of the ranges of real functions.
  - A simple example: $1/3 \in [\downarrow 1 \div 3 \downarrow, \uparrow 1 \div 3 \uparrow] = [0.33...33, 0.33...34]$.

Affine arithmetic

- X in [1 3] =>X = 2 + ε_1, ε_1 in[-1, 1]
- Conversely, X = 5 + 2 ε_2 => X in [3, 7]

Affine Arithmetic (1/2)

- Affine form $\mathbf{x} = x_0 + x_1\varepsilon_1 + \ldots + x_n\varepsilon_n$ (length n)
  - x_i : real **coefficients**
  - $\varepsilon_i \in [-1, 1]$: noise **variables**
- Affine operations $\mathbf{z} \equiv a\mathbf{x} + b\mathbf{y} + c$
  - $\mathbf{z} \equiv (ax_0 + by_0 + c) + \sum(ax_i + by_i)\varepsilon_i$
- Non-affine operations
  - $\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathbf{y}) = f^*(\varepsilon_1,\ldots,\varepsilon_n) \equiv \underbrace{z_0 + z_1\varepsilon_1 + \ldots + z_n\varepsilon_n}_{f^a \text{ is linear}} + \underbrace{z_{\text{new}}\varepsilon_{\text{new}}}_{\text{error bound}}$
  - $\mathbf{z}$ is of length $n + 1$
- The inclusion property:
  - $\forall \mathbf{x} \in \mathrm{A}^n : f(\mathbf{x}) \subseteq \{\mathbf{z} = \mathbf{f}(\mathbf{x}) \mid \forall \varepsilon_i \in [-1, 1]\}$
- **Rounding controls** in floating-point arithmetic
  - Absolute rounding errors are added to the new term $z_{\text{new}}\varepsilon_{\text{new}}$
  - Also allow **rigorous enclosures** of the ranges of real functions

Affine Arithmetic (2/2)

ε's are variables

- Multiplication (two variants):

  ➊ **xy** $= x_0 y_0 + 0.5\sum x_i y_i + \sum(x_0 y_i + y_0 x_i)\varepsilon_i + (0.5\sum|x_i y_i| + \sum_{i\neq j}|x_i y_j|)\ \varepsilon_{\text{new}}$

  complexity $O(n^2)$, **tight enclosure** [Kolev 2001, Messine 1999]

  ➋ **xy** $= x_0 y_0 + 0.5\sum x_i y_i + \sum(x_0 y_i + y_0 x_i)\varepsilon_i + (\sum|x_i|\ \sum|y_i| - 0.5|\sum x_i y_i|)\ \varepsilon_{\text{new}}$

  complexity $O(n)$, but **less tight** than ➊ [Kolev 2002]

Revised Affine Arithmetic (1/2)

- Multiplication :

  ➊ $\mathbf{xy} = x_0 y_0 + 0.5\sum x_i y_i + \sum(x_0 y_i + y_0 x_i)\varepsilon_i + (0.5\sum|x_i y_i| + \sum_{i\neq j}|x_i y_j|)\,\varepsilon_{\text{new}}$

  complexity $O(n^2)$, **tight enclosure** [Kolev 2001, Messine 1999]

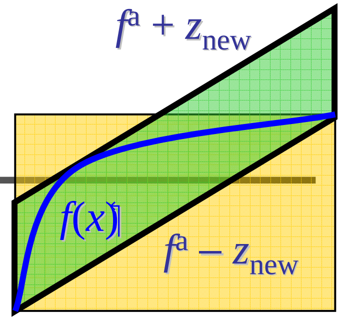
  ➋ $\mathbf{xy} = x_0 y_0 + 0.5\sum x_i y_i + \sum(x_0 y_i + y_0 x_i)\varepsilon_i + (\sum|x_i| \sum|y_i| - 0.5|\sum x_i y_i|)\,\varepsilon_{\text{new}}$

  complexity $O(n)$, but **less tight** than ➊ [Kolev 2002]

- **[Vu 2004] :** the following form has the same number of real operations than ➋, but is as tight as ➊

  $\mathbf{xy} = x_0 y_0 + 0.5\sum x_i y_i + \sum(x_0 y_i + y_0 x_i)\varepsilon_i + (\sum|x_i| \sum|y_i| - 0.5\sum|x_i y_i|)\,\varepsilon_{\text{new}}$

Revised Affine Arithmetic (2/2)



- **Moreover :**
  - $\mathbf{x} = x_1\varepsilon_1 + \ldots + x_n\varepsilon_n + \boxed{x_0 + e_x[-1, 1]} \rightarrow$ (can be replaced with $[l_x, u_x]$)
  - The length will not increase during long-running computations
- [Vu 2004] proposed a **constructive theorem** and a **new generic procedure** to **rigorously** compute Chebyshev affine approximations ($f^a \pm z_{new}$) for monotonously continuously differentiable functions f
  - It needs a **weaker condition than the original** (f is twice continuously differentiable, f'' has the same sign),
  - It can be applied to elementary functions (e.g., $\mathbf{x}^2$, sqrt $\mathbf{x}$, ln $\mathbf{x}$),
  - Affine approximations can be obtained for *factorable* functions by a recursive composition of elementary functions.

Outline

- Problem statement
- **Numerical Constraint Propagation on Dags using a single inclusion representation** (Interval arithmetic)←
- Using multiple inclusion representations on DAGs
  - The CIRD algorithm
- Some experiments
- Conclusions

DAG Representation

By Schichl & Neumaier, 2004

An NCSP

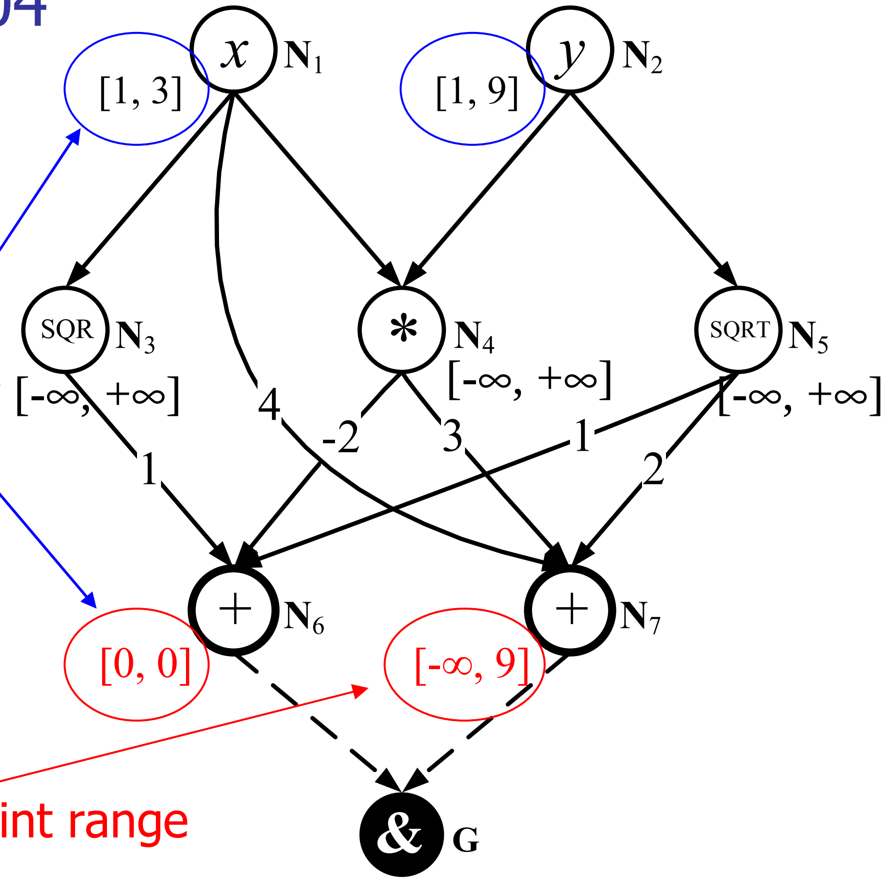
$$\begin{cases} x^2 - 2xy + \sqrt{y} = 0 \\ 4x + 3xy + 2\sqrt{y} \le 9 \\ 1 \le x \le 3 \\ 1 \le y \le 9 \end{cases}$$

$$\begin{cases} x^2 - 2xy + \sqrt{y} \in [0, 0] \\ 4x + 3xy + 2\sqrt{y} \in [-\infty, 9] \\ x \in [1, 3] \\ y \in [1, 9] \end{cases}$$

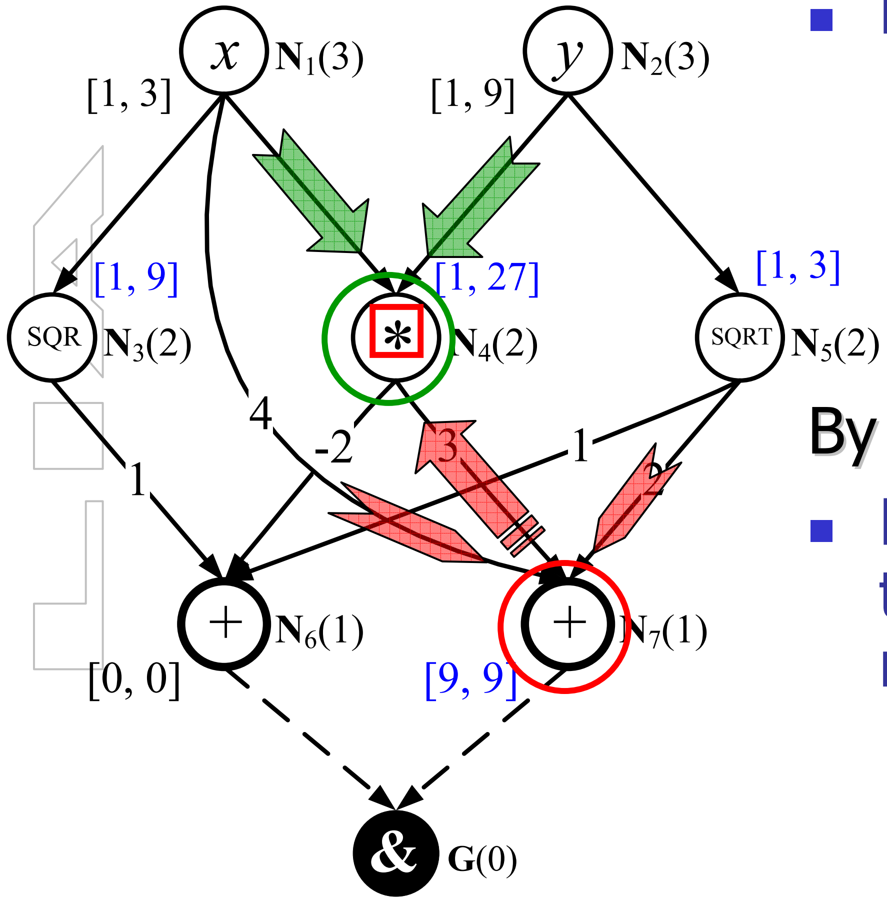
node range

domains

constraint range



Interval Constraint Propagation



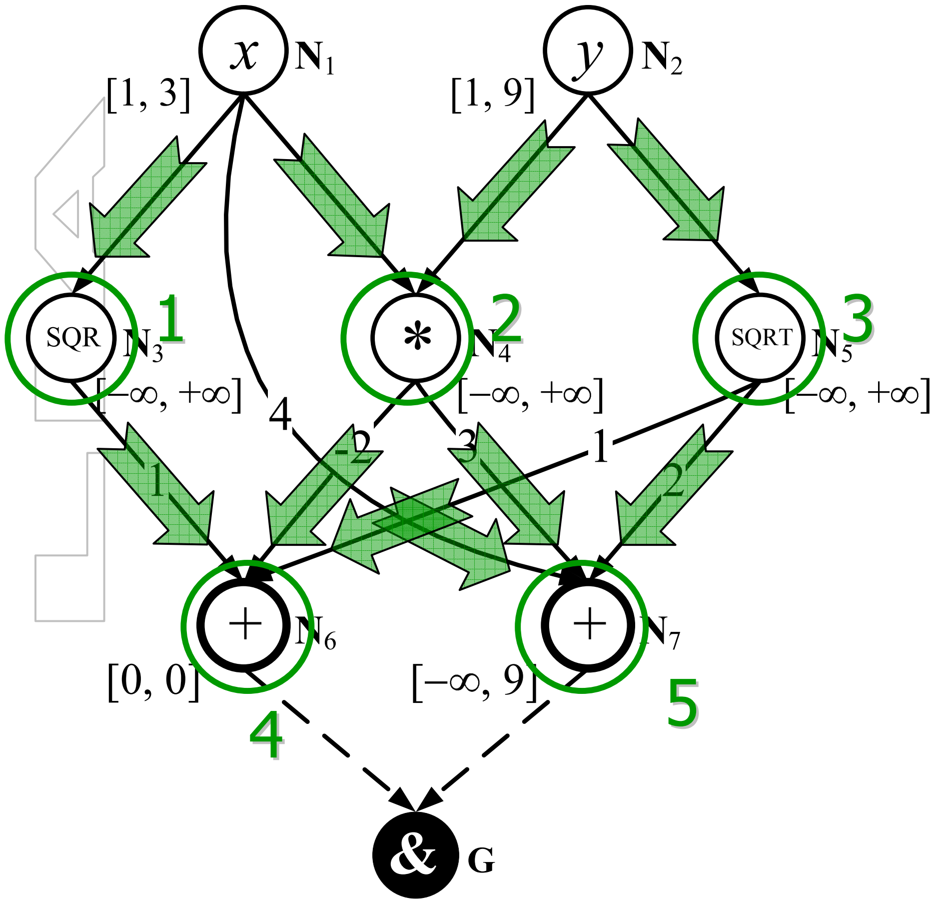
- **Forward Evaluation**: **FE**$(N_4, *)$
  - from $\mathbf{N}_4 = \mathbf{N}_1 * \mathbf{N}_2$,
  - compute $\tau_{N4} := \tau_{N4} \cap (\mathbf{x} * \mathbf{y})$,
  - thus $\tau_{N4} := [-\infty, +\infty] \cap [1, 27] = [1, 27]$

By Benhamou *et al.*, 1999

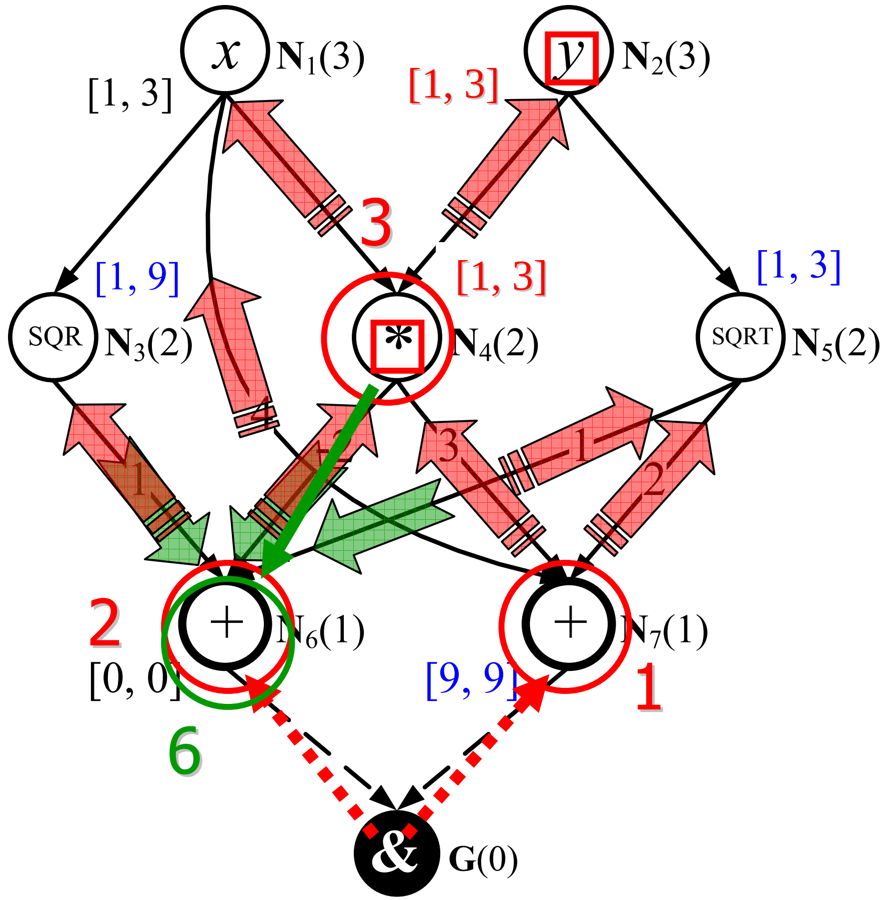
- **Backward Propagation**: **BP**, the approximate projection of a node relation on each child
  - from $\mathbf{N}_7 = 4\mathbf{N}_1 + 3\mathbf{N}_4 + 2\mathbf{N}_5$,
  - write $\mathbf{N}_4 = (\mathbf{N}_7 - 4\mathbf{N}_1 - 2\mathbf{N}_5)/3$,
  - thus $\tau_{N4} := \tau_{N4} \cap (\tau_{N7} - 4\tau_{N1} - 2\tau_{N5})/3 = [1, 27] \cap [-9, 3] = [1, 3]$

Forward-Backward Propagation on DAGs

Recursive Forward Evaluation



Forward-Backward Propagation (FBPD)



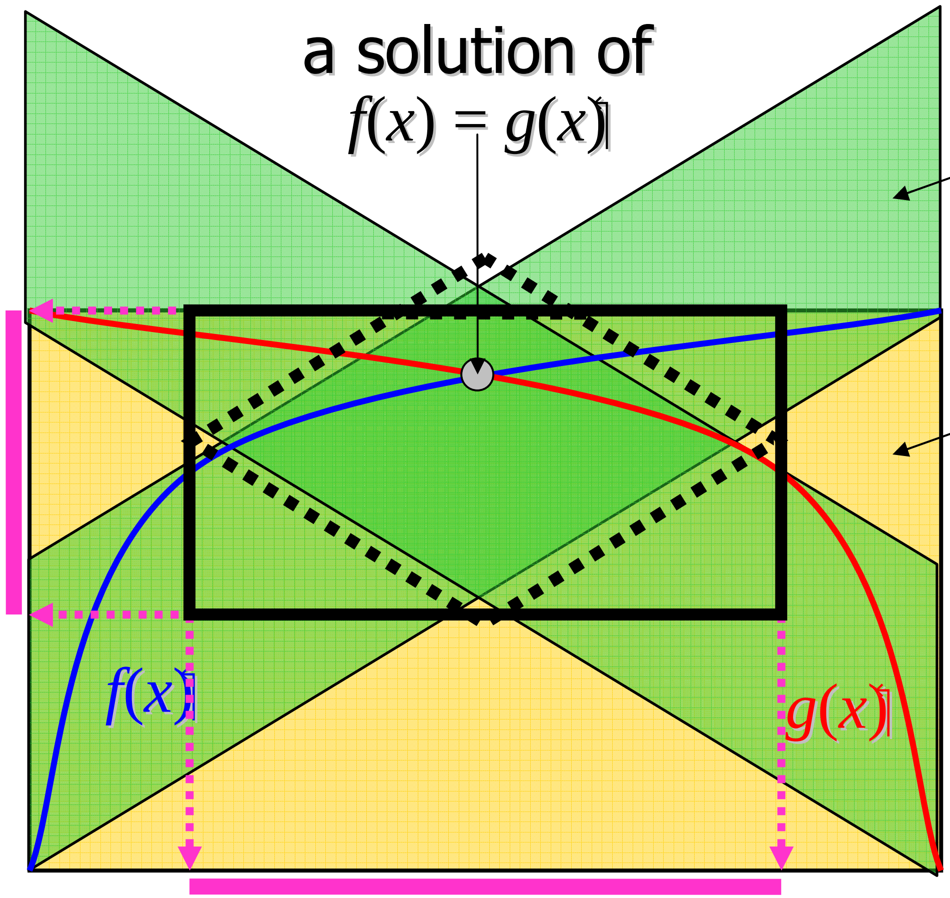
Outline

- Problem statement
- Numerical Constraint Propagation on Dags using a single inclusion representation (Interval arithmetic)
- **Using multiple inclusion representations on DAGs ←**
  - The CIRD algorithm
- Some experiments
- Conclusions

Main idea

Attaching to each node of the DAG redundant inclusion representations in order to get tighter evaluation of its range.

Why combining inclusions?



affine arithmetic

Stolfi *et al.*, 1993

interval arithmetic

Moore *et al.*, 1959

Answer:

to get better effects of domain reduction

safe linear programming (Neumaier & Shcherbina, 2004)

The CIRD Algorithm: building blocks

- Data associated to each node Ni:
  - A set of inclusion representations ($R(N_i)$)
  - A range (interval) ($\tau(N_i)$)

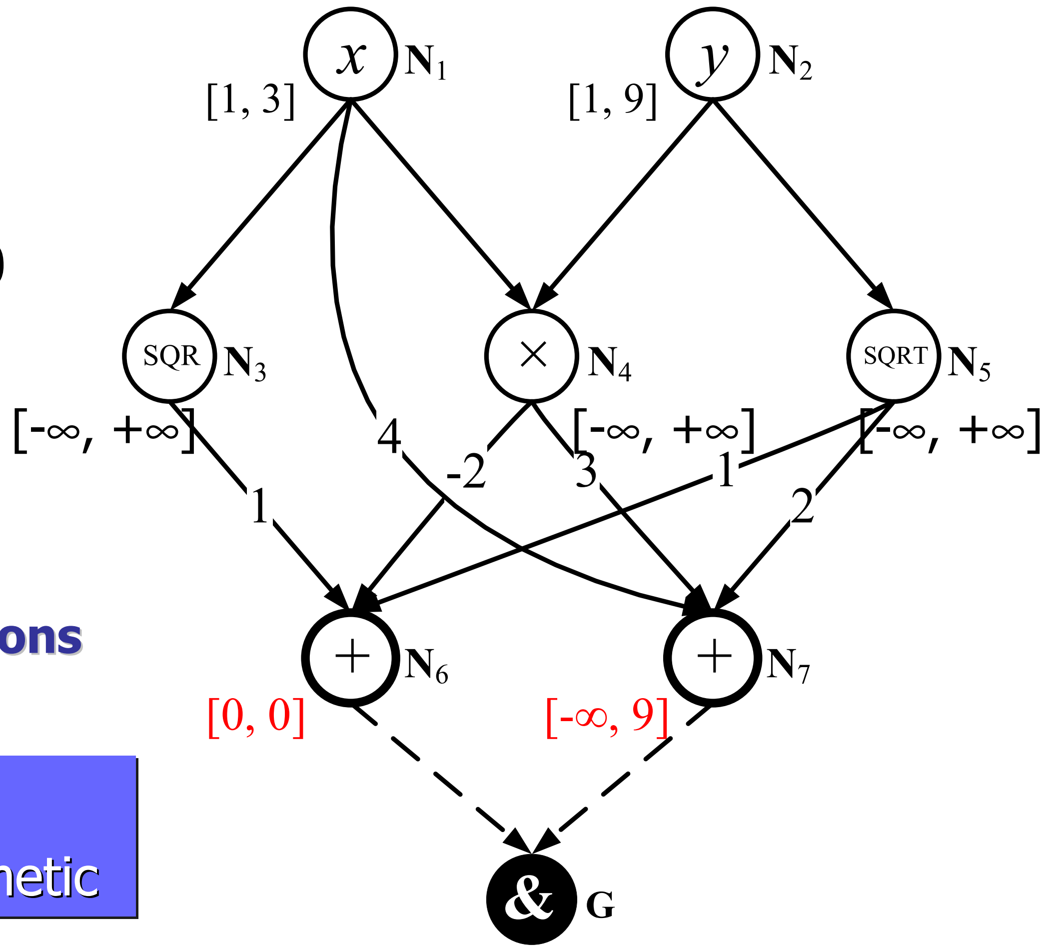
- Node Evaluation:
  - Evaluates the range of the node with respect to each inclusion representation

- Node Pruning:
  - Inclusion Constraints Systems (ICS): the set of redundant constraints that can be inferred from an inclusion representation
  - Pruning Constraint Systems (PCS): all the ICS related to a node + the ICS of its children

Illustrative example: CIRD[ai]

A NCSP:

$$\begin{cases} x^2 - 2xy + \sqrt{y} = 0 \\ 4x + 3xy + 2\sqrt{y} \le 9 \\ 1 \le x \le 3 \\ 1 \le y \le 9 \end{cases}$$

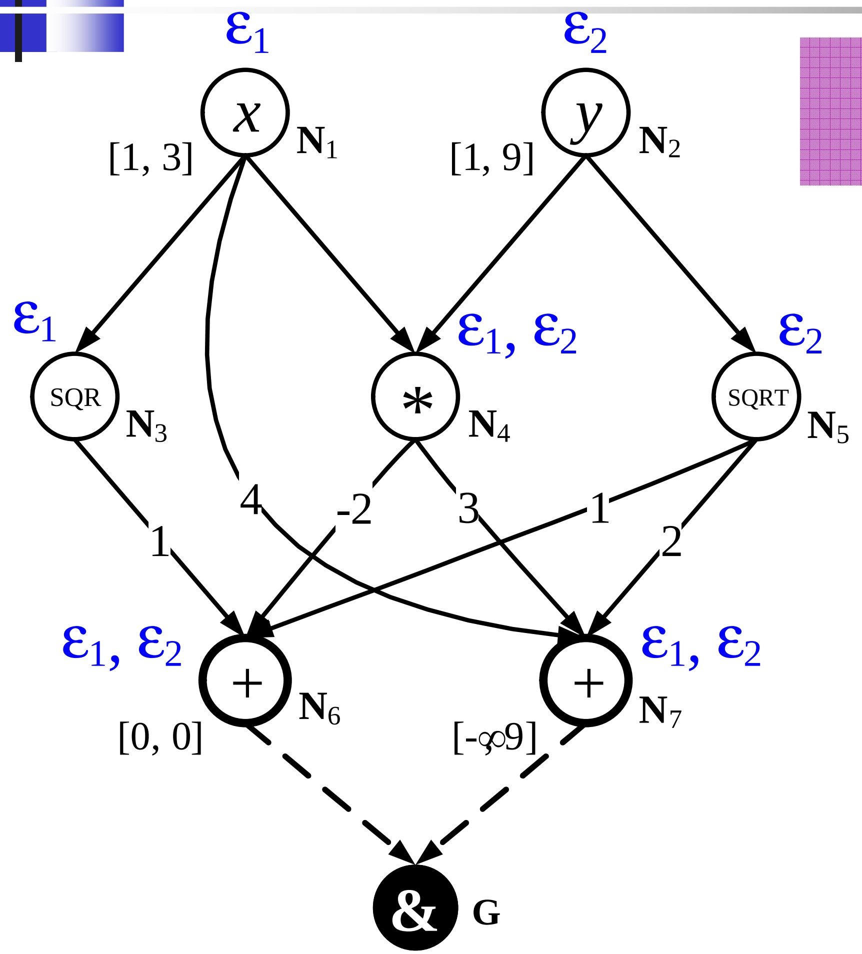


set of inclusion representations

$\Sigma = \{I, A\}$

①	I	interval arithmetic
②	A	revised affine arithmetic

Node Evaluation



Generalization of forward evaluation

$\tau(N_i) = I(N_i) = A(N_i) = [-\infty, +\infty]$, i=3,4,5

$\tau(N_1) = I(N_1) = [1, 3]$, $A(N_1) = 2 + \varepsilon_1$

$\tau(N_2) = I(N_2) = [1,9]$, $A(N_2) = 5 + 4\,\varepsilon_2$

$I(N_4) = I(N_1) * I(N_2)$
$= [1,27]$

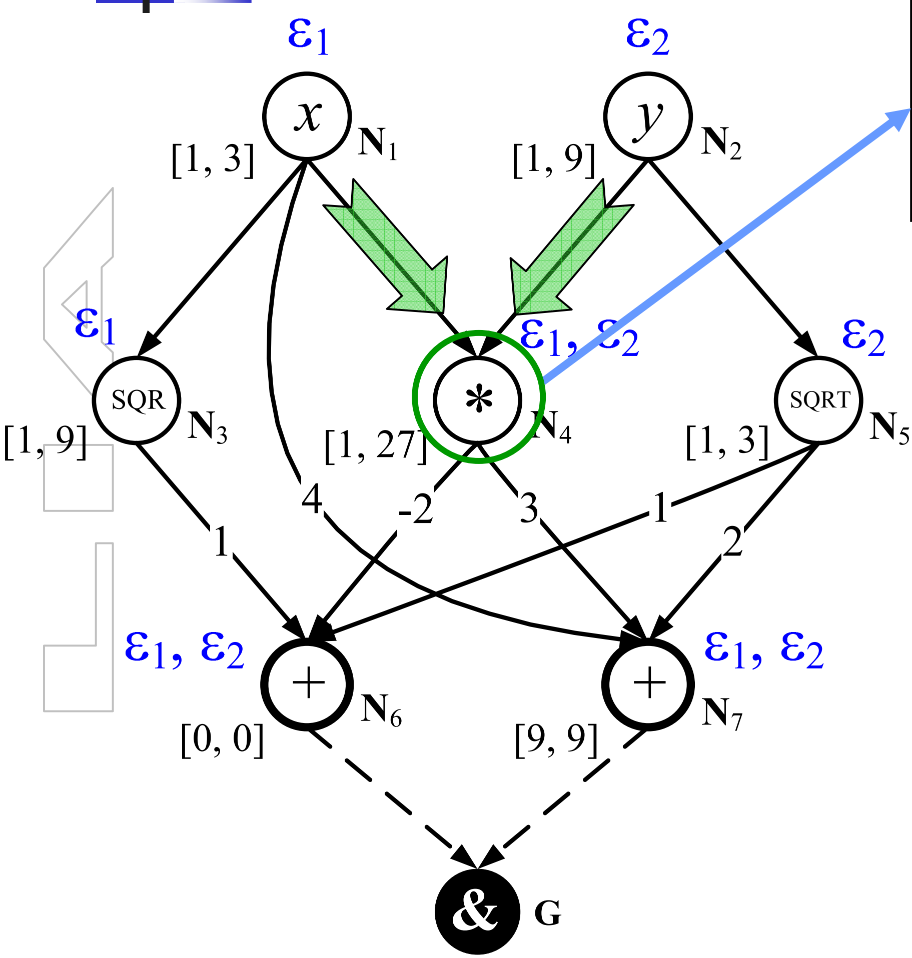
$\tau(N_4) = \tau(N_4) \cap \mu(I(N_4))$
$= [1, 27]$

$A(N_4) = A(N_1) * A(N_2)$
$= 10 + 5\varepsilon_1 + 8\varepsilon_2 + 4[-1, 1]$

$\tau(N_4) = t(N_4) \cap \mu(A(N_4))$
$= [1, 27]$

μ : interval evaluation of the inclusion representation

Inclusion Constraint Systems (ICS)



Data at N_4:
- $I(N_4) = [1, 27]$
- $A(N_4) = 10 + 5\varepsilon_1 + 8\varepsilon_2 + 4[-1, 1]$
- $\tau(N_4) = [1, 27]$

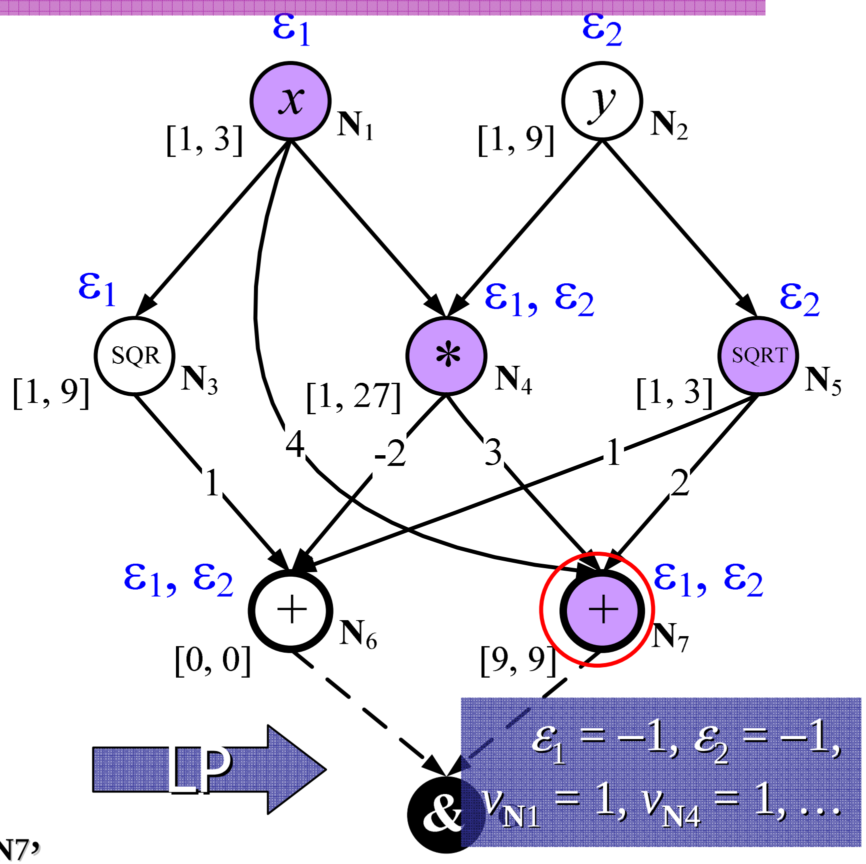
- **ICS**$(I(N_4), \tau(N_4))$:
  - $v_{N4} \in [1, 27]$
- **ICS**$(A(N_4), \tau(N_4))$:
  - $10 + 5\varepsilon_1 + 8\varepsilon_2 + 4\varepsilon_{N4} = v_{N4}$
  - $v_{N4} \in [1, 27]$
  - $(\varepsilon_1, \varepsilon_2, \varepsilon_{N4}) \in [-1, 1]^3$

Pruning Constraint Systems (PCS)

PCS(N_7,…) = all inclusion constraint systems at N_7 and its children

Don't read the details

- **PCS(N_7, {I}):** less informative
  - $4v_{N1} + 3v_{N4} + 2v_{N5} = v_{N7}$,
  - $v_{N1} \in [1, 3]$, $v_{N4} \in [1, 27]$,
  - $v_{N5} \in [1, 3]$, $v_{N7} \in [9, 9]$.
- **PCS(N_7, {A}):** more informative
  - $4v_{N1} + 3v_{N4} + 2v_{N5} = v_{N7}$,
  - $v_{N1} \in [1, 3]$, $v_{N4} \in [1, 27]$,
  - $v_{N5} \in [1, 3]$, $v_{N7} \in [9, 9]$,
  - $2 + \varepsilon_1 = v_{N1}$,
  - $10 + 5\varepsilon_1 + 8\varepsilon_2 + 4\varepsilon_{N4} = v_{N4}$,
  - $2.125 + \varepsilon_2 + 0.125\varepsilon_{N5} = v_{N5}$,
  - $42.25 + 19\varepsilon_1 + 26\varepsilon_2 + 12.25\varepsilon_{N7} = v_{N7}$,
  - $(\varepsilon_1, \varepsilon_2, \varepsilon_{N4}, \varepsilon_{N5}, \varepsilon_{N7}) \in [-1, 1]^5$.



The CIRD algorithm – main steps

Intialization Phase:

- Initial recursive node evaluation
- Initialization of two Waiting Lists : L_e, the list of nodes waiting for evaluation, and L_p, the list of nodes waiting for pruning

Propagation Phase: repeat until both L_e and L_p become empty or the limit, if any, on the number of iterations is reached:

Get the next node N according to some strategy:

- From L_p first (pruning-first strategy) until it becomes empty
- From one of the two (in a rotationnal way)
- …

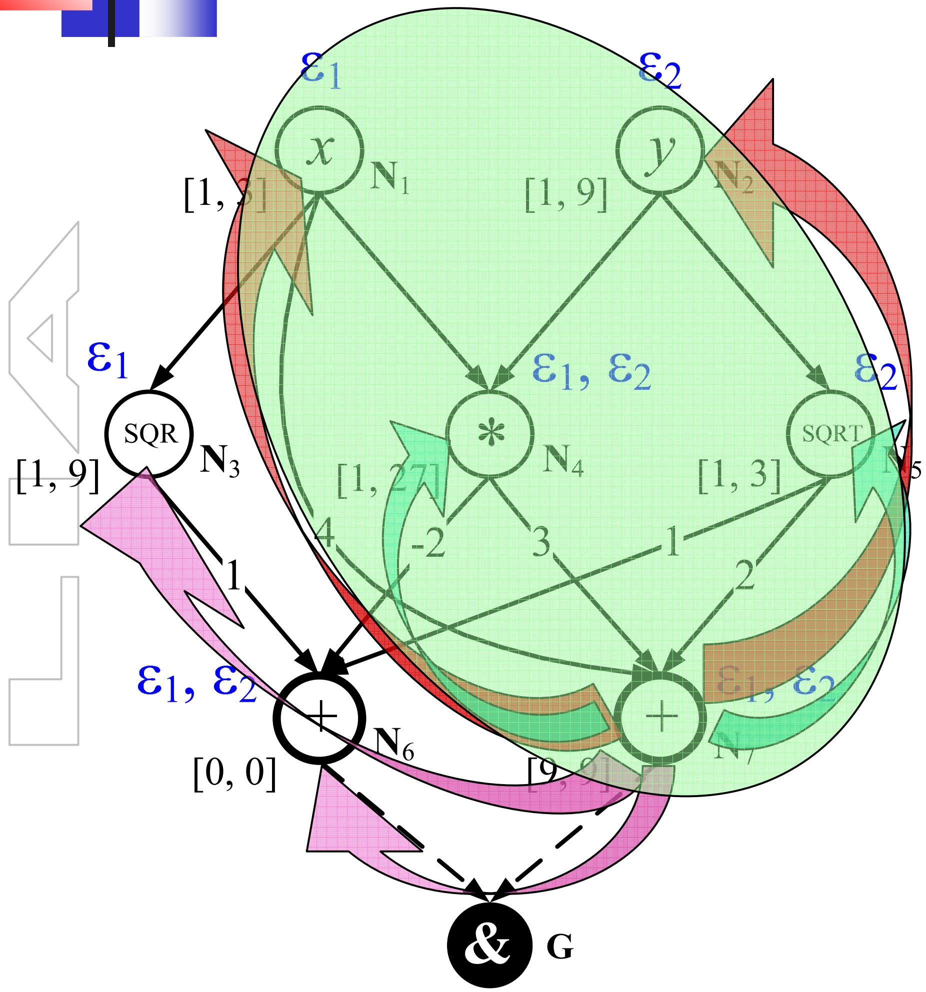
if N was taken from L_e, perform Node Evaluation on N

- If this returns an empty set, the algorithm terminates with an infeasible status.
- If the changes of τ(N) is considered enough, put each parent (if any) of N in L_e and put N in L_p

else perform Node pruning : use dedicated pruning techniques on the PCSs related to N, to generate a new range for N

- If this process returns an empty set, the algorithm terminates with an infeasible status
- else update the ranges of the related nodes
- For each of these nodes, M, if the changes of τ(M) is considered enough, put each parent (if any) of M in L_e and put M in L_p

Node Range Updates



- Prune **PCS**$(N_7, \{A\})$ using LP, we get
  - $\boxed{\varepsilon_1 = -1, \varepsilon_2 = -1}$
  - optional: $v_{N1} = 1, v_{N4} = 1, \ldots$
- **Leaf Update**: update only the leaves
  $x := 2 + \varepsilon_1 = 1$
  $y := 5 + 4\varepsilon_2 = 1$
- **Child Update**: update only the children like in the backward propagation
- The combination of them
- Update all nodes with reduced auxiliary variables (ε_i)
- Update only descendants

The CIRD algorithm

For a formal presentation of the algorithm, see:

« Rigorous Solution Techniques for Numerical Constraint Satisfaction Problems »

Thesis # 3155, 2005

Author: Xuan-Ha Vu

Swiss Federal Institute of Technology, Lausanne

http://liawww.epfl.ch/Publications/Archive/vxhthesis.pdf

Test cases

- T1 : 8 easy problems with isolated solutions
- T2: 4 average problems with isolated solutions
- T3: 8 hard problems with isolated solutions
- T4: 7 easy problems with continuum of solutions
- T5: 8 hard problems with continuum of solutions

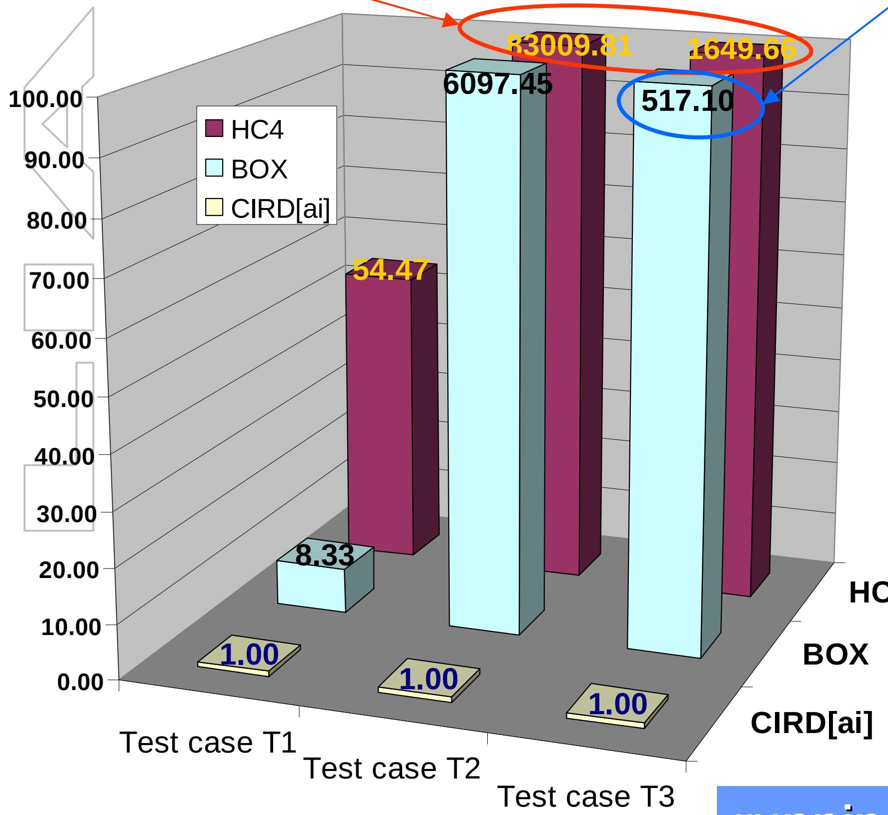
Test criteria

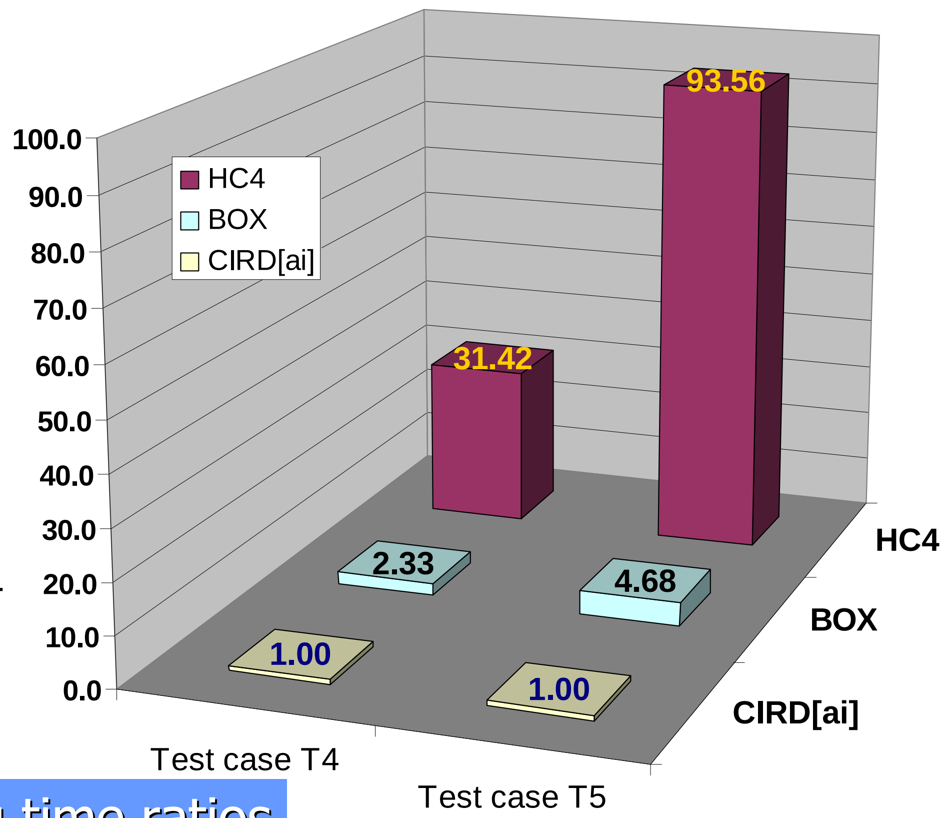
- Relative time ratio : running time
- Relative cluster ratio : number of boxes in the output
- Relative iteration ratio : number of splits
- Relative reduction ratio : $(V / D)^{1/d}$ (V = volume of the output, D= volume of the original domain, d = dimension of the problem)
- Inner volume ratio (ratio of the volume of inner boxes to the volume of output boxes)

Experiments – General Techniques

NCSPs with isolated solutions | continuums of solutions

timeout **10h**, < 10^6 splits | 10^6 splits without timeout

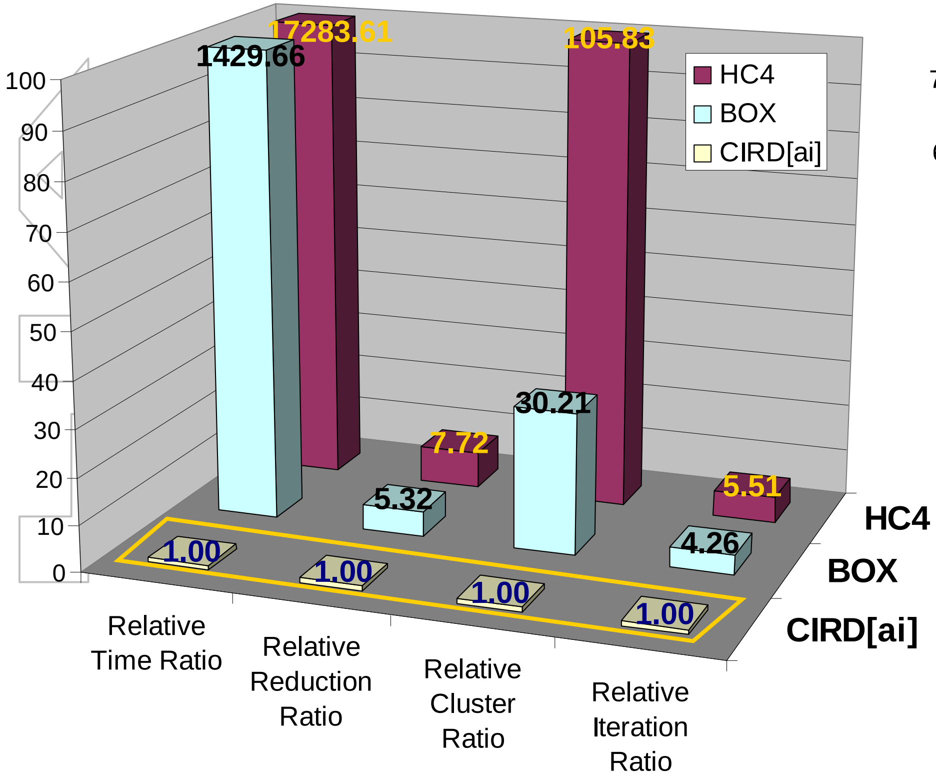




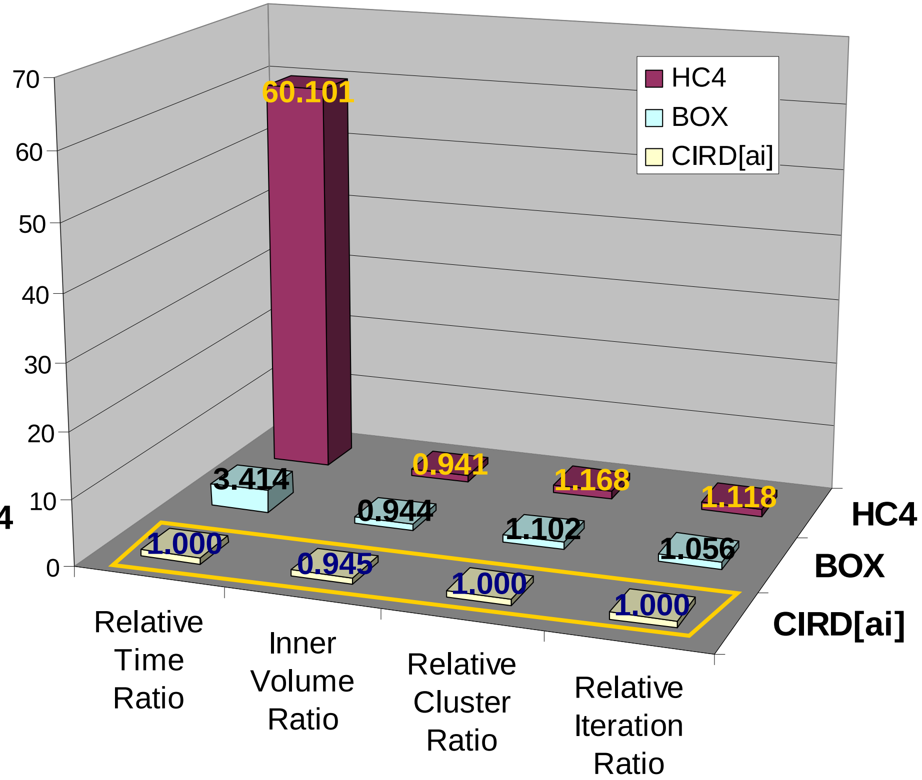
running time ratios

Experiments – Summary

NCSPs with isolated solutions



continuum of solutions



Experiments – Other Techniques

Some other preliminary comparisons:

- **CIRD[ai]** ≈ 30 times faster than Kolev's technique (**A2**) for the benchmark in [Kolev, 2002]
  - a mathematical technique using affine arithmetic,
  - without guaranteed rigor, require some posterior assumptions;
  - **CIRD and A2 should be collaborative rather than competitive**:
    - the reduction rule in **A2** can be used in place of LP in **CIRD[ai]**.
- **CIRD[ai]** ≈ 10–40 times faster than **Quad** for two benchmarks in [Lebbah *et al.*, Aug 2003, Nov 2003]
  - a linear relaxation based filtering technique with guaranteed rigor.
  - **CIRD and Quad should be collaborative rather than competitive**:
    - **Quad** can be used in **CIRD** to tackle the quadratic form [Messine 1999] and power operations x^n.

Conclusion

- CIRD is intended to be a **generic scheme** for combining multiple inclusion techniques in numerical constraint propagation:
  - users can devise their own combination strategies, depending on the set of inclusion representations
- We studied **CIRD[ai]**, **an instance of the CIRD** scheme combining revised affine arithmetic with interval arithmetic
  - Some potential

Currently on the agenda

- Replacement of linear programming by less costly domain reduction techniques
- Integration of Kolev generalized affine arithmetic
- Integration of linear relaxation techniques (eg. [Borradaile & Van Hentenryck 2004]
- Investigate the integration of higher-order inclusion techniques (convexification)
- Comparison with other approaches [Granvilliers & Benhamou 2006]
- …

Thank you for your attention