

Symbolic Execution with Interval Constraint Solving and Meta-Heuristic Search

Mateus Borges, Marcelo d'Amorim (*Fed. U. Pernambuco, Brazil*)

Saswat Anand (*Georgia Inst. of Technology*)

David Bushnell, Corina Pasareanu (*TRACLabs and CMU/NASA Ames*)

April, 2012

Software errors are expensive

Software is everywhere



Annual cost of software errors to the US economy is ~60B dollars [NIST2002]

Main approaches for finding errors

- Software Testing
 - Run code and observe effects
- Static Analysis
 - Analyze code without running it

Research Goal

Develop new techniques and improve existing techniques for Software Testing and Static Analysis

Symbolic Execution (SE)

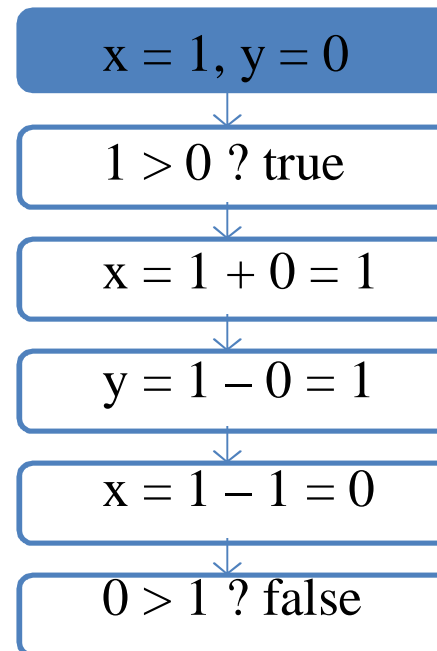
- Analysis of programs with unspecified inputs
 - Execute a program on symbolic inputs
 - Introduced in the 70s by L. Clarke, J. King
- Symbolic states represent sets of concrete states
- For each path, build a **path condition**
 - Condition on inputs for execution to follow that path
 - Check **satisfiability of path condition**
 - Explore only feasible paths
- Symbolic state
 - Symbolic values/expressions for variables, path condition, and program counter

Example: Standard Execution

Code that swaps 2 integers

```
int x, y;  
if (x > y) {  
    x = x + y;  
    y = x - y;  
    x = x - y;  
if (x > y)  
    assert false;  
}
```

Concrete Execution Path

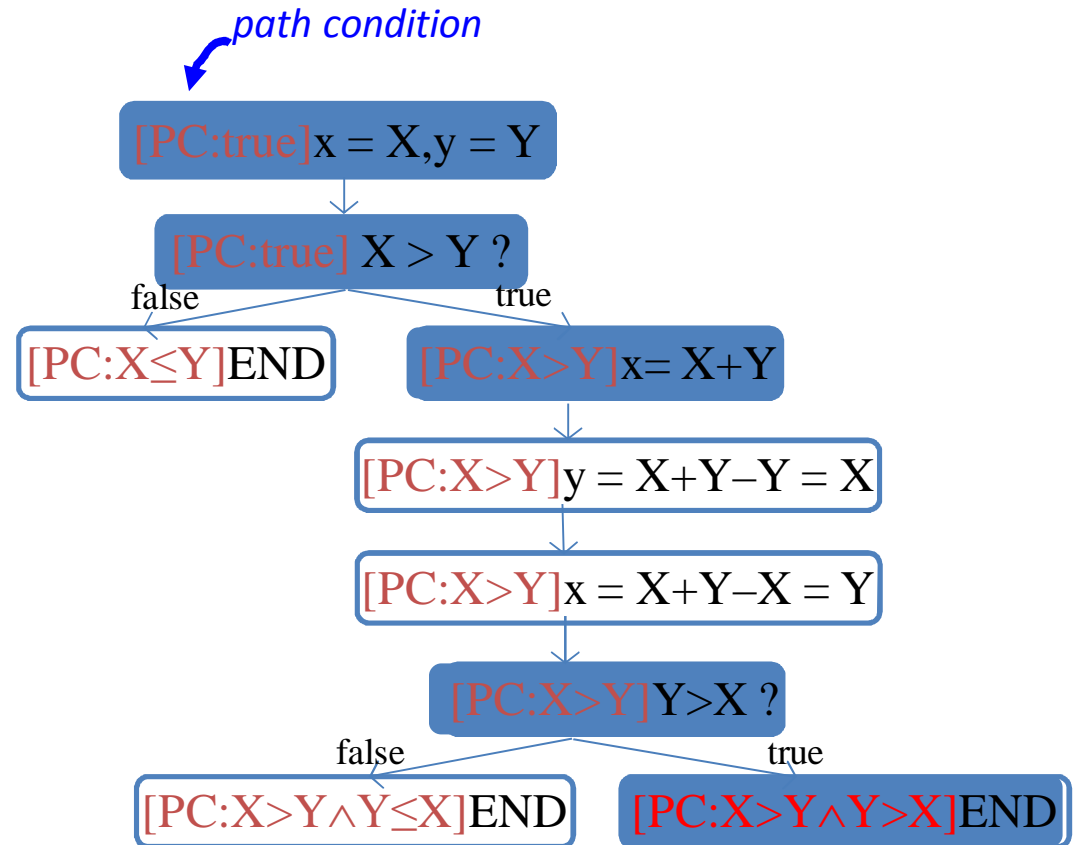


Example: Symbolic Execution

Code that swaps 2 integers:

```
int x, y;  
if (x > y) {  
    x = x + y;  
    y = x - y;  
    x = x - y;  
    if (x > y)  
        assert false;  
}
```

Symbolic Execution Tree:



False!

Solve path conditions → test inputs

State-of-the-Art

- Explosion in number of tools and techniques
 - **Tools:** CUTE, CREST, DART, EGT, EXE, jCUTE, KLEE, PEX, PREFIX, SAGE, SMART, SPF, YOGI, etc.
 - **Industry:** Bell Labs, Fujitsu, IBM, Microsoft, etc.
 - **Government Labs:** NASA, etc.
 - **Universities:** Berkeley, EPFL, Illinois, Imperial College London, Iowa State, U Nebraska Lincoln, UT Austin, UT Arlington, Stanford, Stellenbosch, etc.
 - **Research communities:** software engineering, programming languages, systems, security, etc.

Challenge

Handling complex mathematical constraints

Example constraint generated with SE
for a module from TSAFE (Tactical
Separation Assisted Flight Environment)



```
sqrt(pow(((x1 + (e1 * (cos(x4) - cos((x4 + (((1.0 * (((c1 * x5) *  
(e2/c2))/x6)) * x2)/e1)))))) - (((e2/c2)) * (1.0 - cos((c1 * x5))))),2.0))  
> 999.0 & (c1 * x5) > 0.0 & x3 > 0.0 & x6 > 0.0 & c1 = 0.017... &  
c2 = 68443.0 & e1 = ((pow(x2,2.0)/tan((c1*x3)))/c2) &  
e2 = pow(x6,2.0)/tan(c1*x3)
```

CORAL: CONSTRAINT SOLVER FOR COMPLEX CONSTRAINTS

CORAL

- Target application of solver: SE of programs that
 - Use floating-point arithmetic
 - Call specific math functions

TSAFE example



Input: `sqrt(pow(((x1 + (e1 * (cos(x4) - ...`

Output: `{x1=100.0, x2=98.48..., x3=3.08...E-11, ...}`

Approach: combine meta-heuristic search and interval solving

Meta-Heuristic Search

- Explores candidate solutions
 - Start with random solutions
 - Refine candidate set based on *fitness function*

Inherently incomplete!

Meta-Heuristic Search

- Local search: Uses **one** single candidate solution
 - E.g., Alternating Variable Method (AVM), hill climbing, simulated annealing, etc.
- Global search: Uses **several** candidate solutions
 - E.g., Particle Swarm Optimization (PSO), genetic algorithms, etc.

Interval Solving

- Another method of constraint solving
- Applications in several domains. E.g., stability in control systems.

Input: $\text{sqrt}(\text{pow}(((x1 + (e1 * (\cos(x4) - \dots$

Output: $\{x1=[\underline{99.9\dots}, 100.0], x2=[99.9\dots, 100.0], \dots\}, \dots$

interval

Intervals may not
contain solutions!

Our Approach: Combine Techniques

Meta-heuristic search

- + Good for finding exact solutions in large search spaces
- May get lost in local maxima

Interval solving

- + Good for computing parts of solution space
- Does not compute solutions


Seed meta-heuristic search
with inputs drawn from intervals
(intuition: better initial states)

Implementation

- For meta-heuristic search
 - Uses Opt4J (<http://opt4j.sourceforge.net/>)
- For interval solving
 - Uses RealPaver (RP) [Granvilliers & Benhamou, 2006]
- Also includes several optimizations
 - E.g., inference of variable domains, elimination of dependent variables, etc.

Tool

- Integrated with NASA's Symbolic PathFinder (SPF)
<http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/jpf-symbc>
- Also available as a library
<http://pan.cin.ufpe.br/coral>



The image shows a screenshot of the CORAL website. At the top, there is a dark grey header with the word "coral" in orange lowercase letters, followed by the tagline "randomized constraint solvers" in white lowercase letters. Below the header is a navigation menu with four items: "Home", "Documentation", "Download", and "Publications", each in a light grey box. The main content area below the menu has the heading "What is CORAL?" and a paragraph describing CORAL as a meta-heuristic constraint solver for numerical constraints in mathematical functions. Below that is the heading "Target" and a paragraph explaining the goal of CORAL to improve symbolic execution of numeric applications.

coral
randomized constraint solvers

Home | Documentation | Download | Publications

What is CORAL?

CORAL is a meta-heuristic constraint solvers for dealing with numerical constraints in mathematical functions.

Target

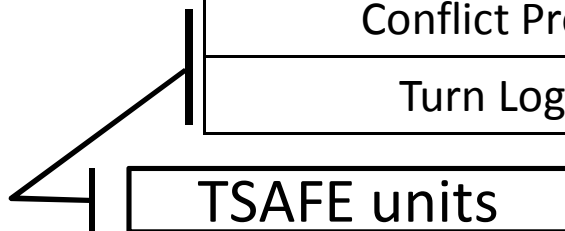
The goal of CORAL is to improve symbolic execution of numeric applications. Symbolic generate test input data. It requires a constraint solver component to solve the cons program. Certain classes of constraints admit a (decision) procedure that can determ

EVALUATION

Subjects

- Publicly available applications from the aerospace domain

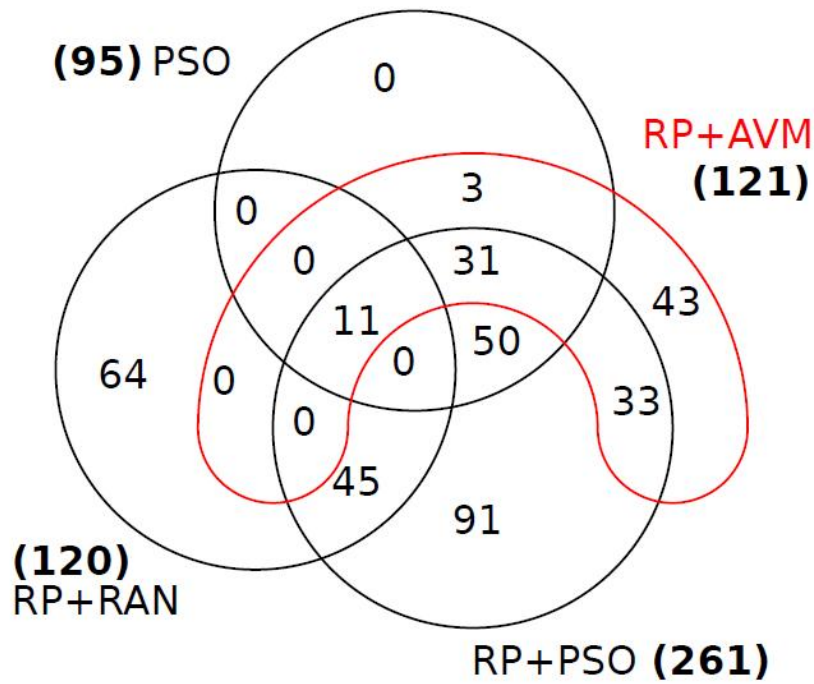
| Subject | # constraints | # conjuncts | # functions |
|---------------------------|---------------|-------------|-------------|
| Apollo Autopilot | 800 | 39 | 3 |
| Collision Detection (CDx) | 800 | 63 | 6 |
| Conflict Probe | 33 | 7 | 5 |
| Turn Logic | 329 | 3 | 20 |



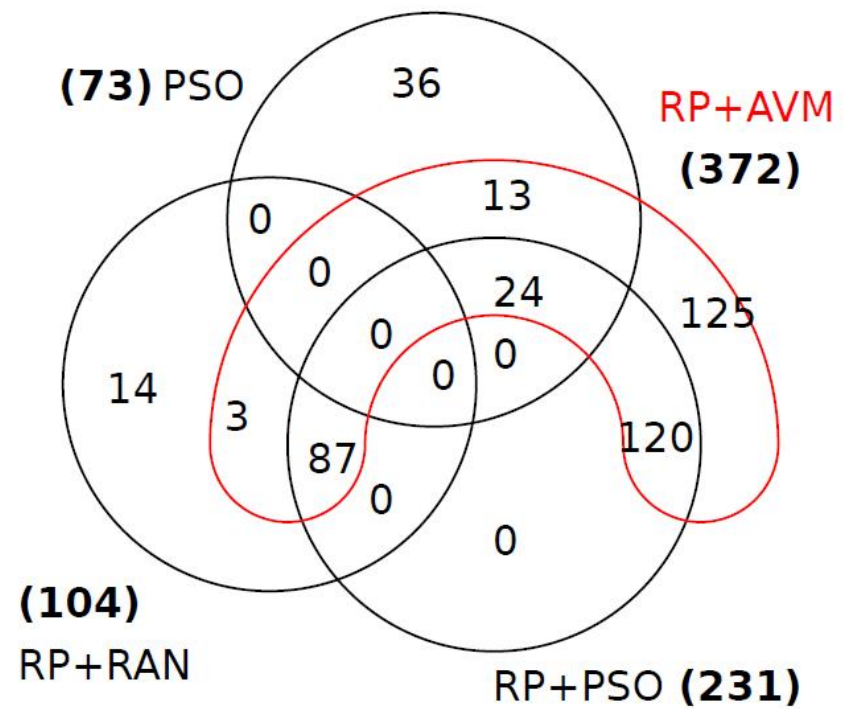
Evaluated CORAL Configurations

- Meta-heuristic search alone
 - AVM
 - PSO (previously found it better than GA)
- Interval solving w/ RealPaver (RP) alone
 - RP+RAN (choose random values from interval)
- Combinations of IS with global and local search
 - RP+AVM – optimistic on RP reported intervals
 - RP+PSO – not so optimistic

Results for Apollo and CDx

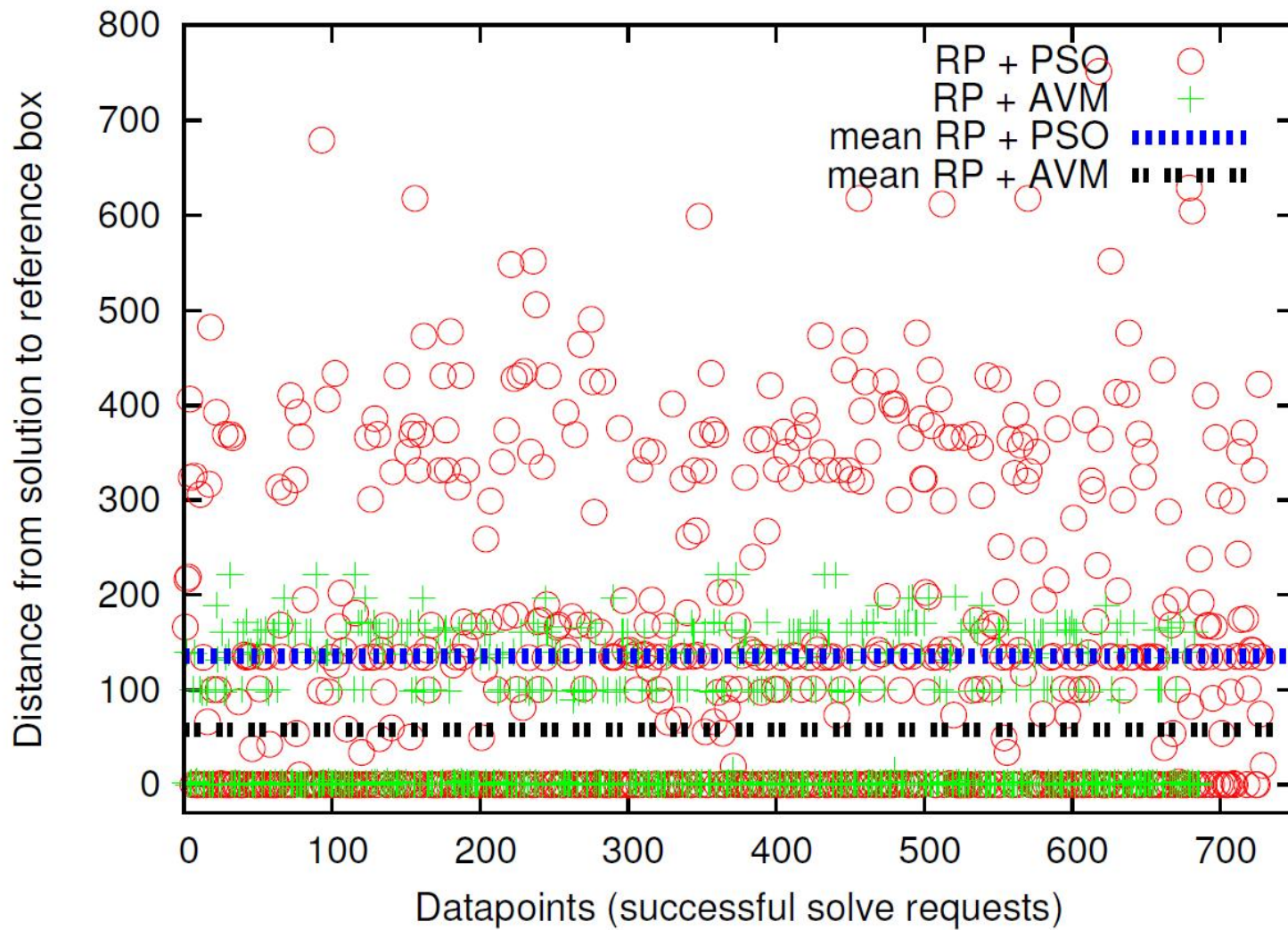


APOLLO

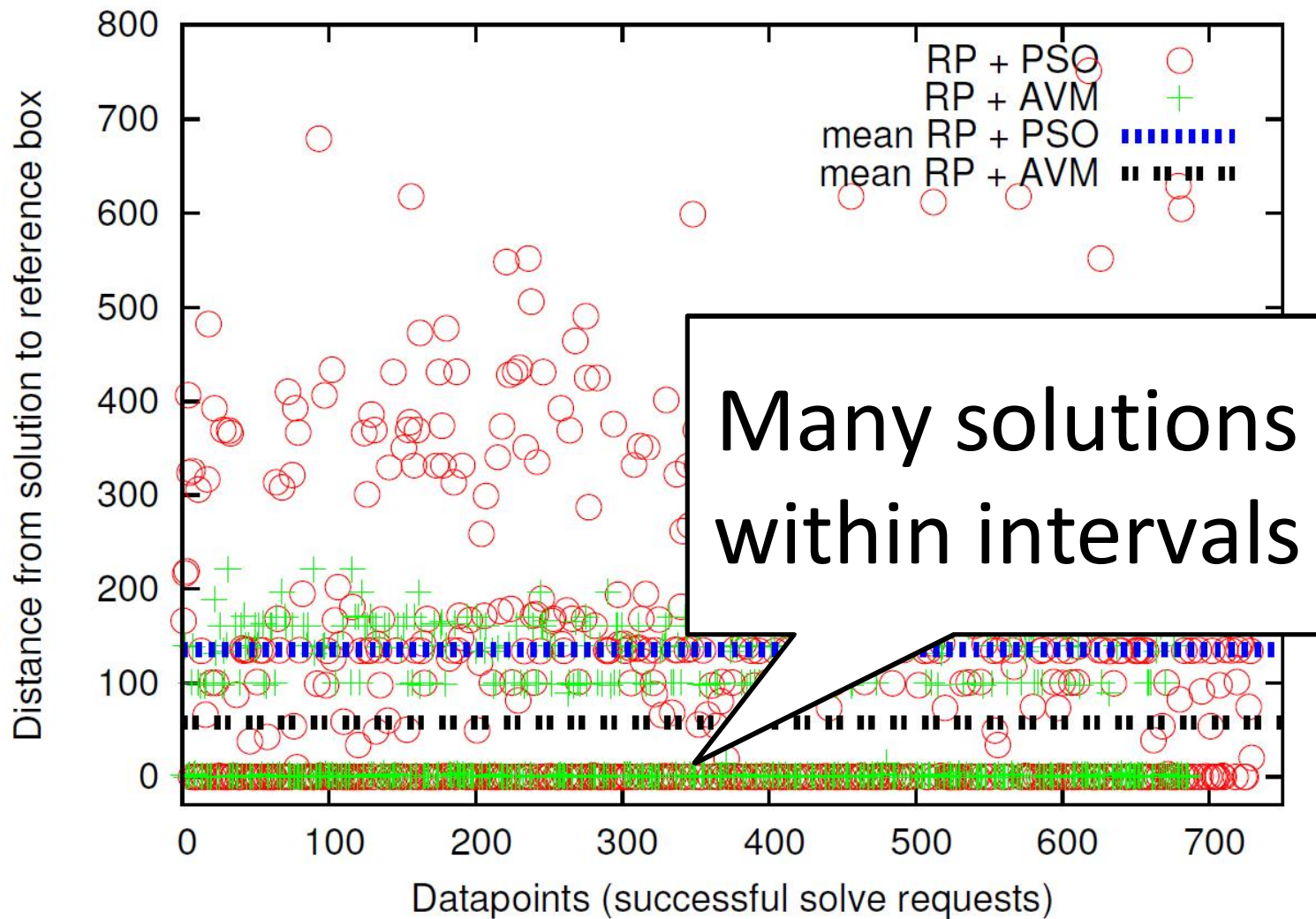


CDx

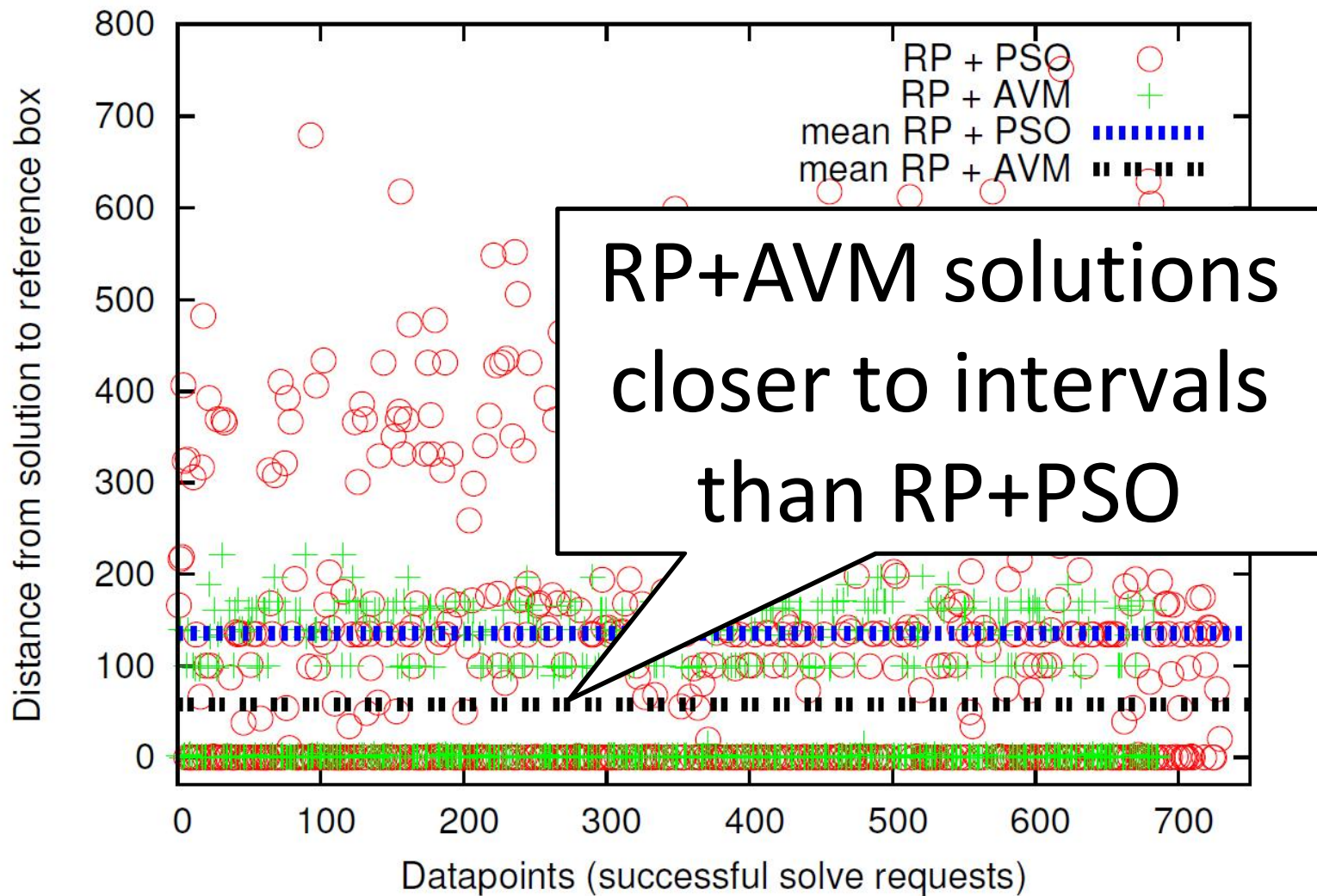
Distance from Solution to RP Box



Distance from Solution to RP Box



Distance from Solution to RP Box




Time Results

- Solvers typically respond very fast
- The most extreme case: ~40s
- We limit the # of iterations not execution time
- Random solving slower than others (10^6 iterations vs 600 PSO)

Conclusions for CORAL

- Combination solved more constraints than meta-heuristic search or interval solving alone
 - Both global and local search help interval solving
 - Complementary: should be run together in parallel

<http://pan.cin.ufpe.br/coral>



The screenshot shows the top portion of the CORAL website. At the top, the word "coral" is written in a bold, orange, lowercase font. Below it, the text "randomized constraint solvers" is displayed in a smaller, grey, lowercase font. A horizontal navigation bar contains four items: "Home", "Documentation", "Download", and "Publications", each enclosed in a light grey box with a thin border. Below the navigation bar, the heading "What is CORAL?" is followed by a paragraph: "CORAL is a meta-heuristic constraint solvers for dealing with numerical constraints in mathematical functions." Below this, the heading "Target" is followed by another paragraph: "The goal of CORAL is to improve symbolic execution of numeric applications. Symbolic generate test input data. It requires a constraint solver component to solve the cons program. Certain classes of constraints admit a (decision) procedure that can determ".

coral
randomized constraint solvers

Home Documentation Download Publications

What is CORAL?

CORAL is a meta-heuristic constraint solvers for dealing with numerical constraints in mathematical functions.

Target

The goal of CORAL is to improve symbolic execution of numeric applications. Symbolic generate test input data. It requires a constraint solver component to solve the cons program. Certain classes of constraints admit a (decision) procedure that can determ

Related Work

- Combining interval solving and SAT solving.
 - iSAT [Franzle *et al.*, JSAT'10]
- Using meta-heuristic search.
 - FLoPSy [Lakhotia *et al.*, ICTSS'10]
- Optimizing solving with constraint propagation.
 - Choco [Jussien *et al.*, EMNTR'10]
- Use concrete values and randomization to simplify complex constraints.
 - DART [Godefroid *et al.*, PLDI'05],
 - EXE [Cadar *et al.*, CCS'06]

Conclusions

- CORAL
 - Solver for complex mathematical constraints generated by symbolic execution
 - Combines meta-heuristic search with interval solving
 - Techniques are effective and complementary
- Future Work
 - Robustness analysis [Majumdar & Saha, RTSS 2009]
 - Overflow and round-off error bounds analysis

THANK YOU!