Interval Constraint Propagation Results

Group:

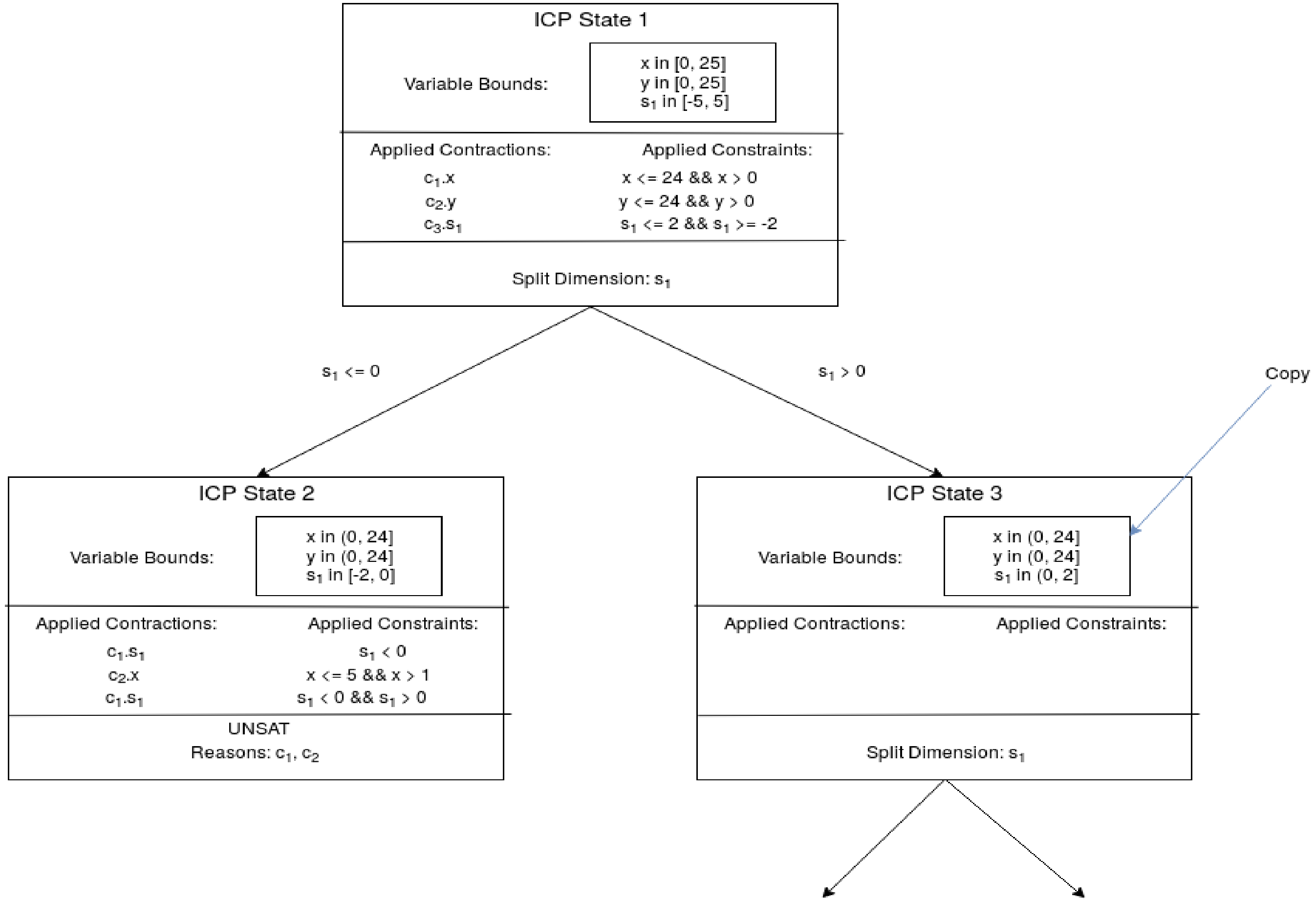
Vincent Drury

Konstantin Perun

David Wlazlo

ICP Search Tree

- Variable Bounds Copies, Independent Nodes → Easy Multi-Threading



ICP State 1

Variable Bounds: x in [0, 25] y in [0, 25] s_1 in [-5, 5]

Applied Contractions: $c_1.x$ $c_2.y$ $c_3.s_1$

Applied Constraints: x <= 24 && x > 0 y <= 24 && y > 0 s_1 <= 2 && s_1 >= -2

Split Dimension: s_1

s_1 <= 0

s_1 > 0

Copy

ICP State 2

Variable Bounds: x in (0, 24] y in (0, 24] s_1 in [-2, 0]

Applied Contractions: $c_1.s_1$ $c_2.x$ $c_1.s_1$

Applied Constraints: s_1 < 0 x <= 5 && x > 1 s_1 < 0 && s_1 > 0

UNSAT

Reasons: c_1, c_2

ICP State 3

Variable Bounds: x in (0, 24] y in (0, 24] s_1 in (0, 2]

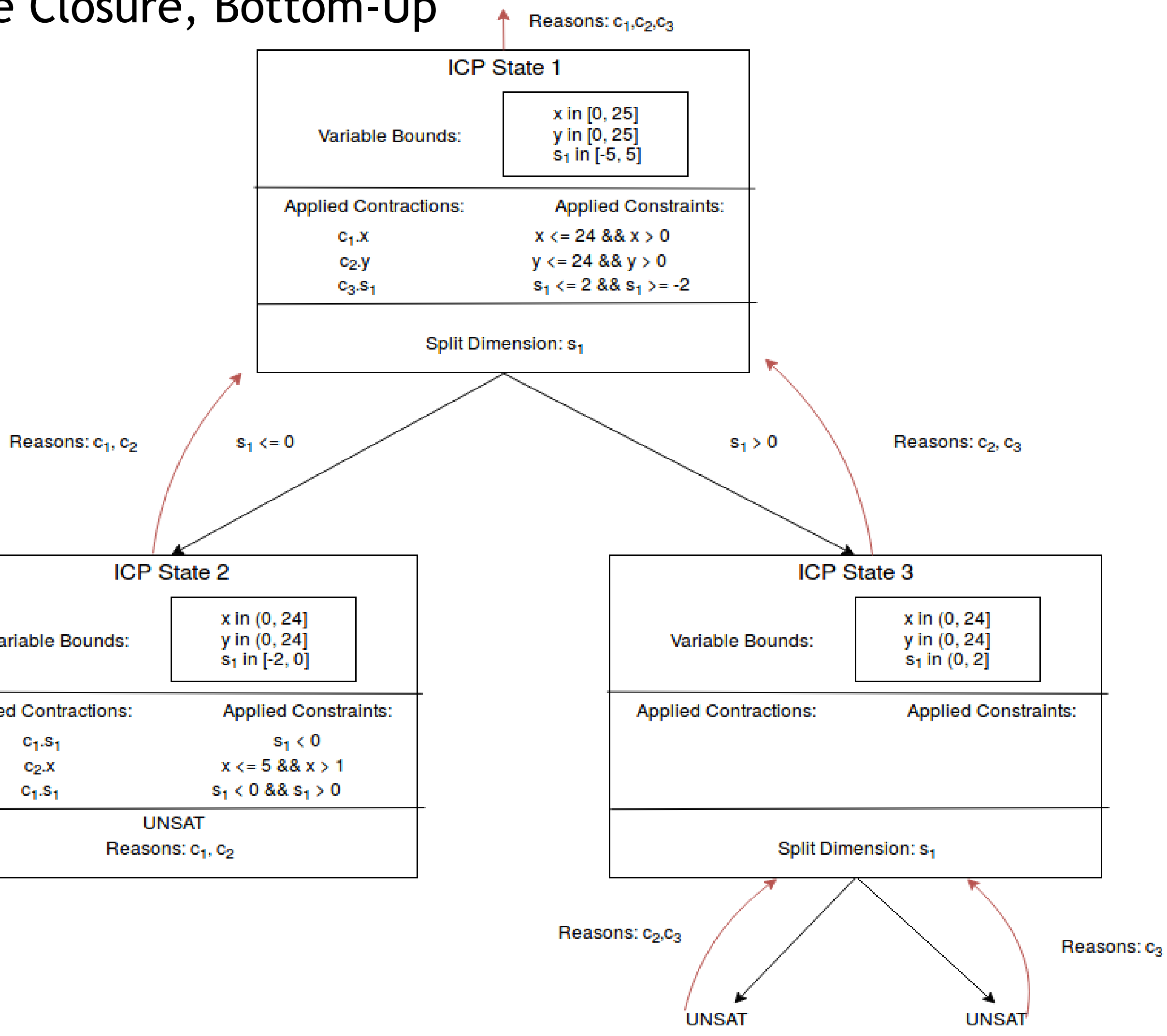
Applied Contractions:

Applied Constraints:

Split Dimension: s_1

UNSAT Cores

- Transitive Closure, Bottom-Up



Reasons: c_1,c_2,c_3

ICP State 1

Variable Bounds:

x in [0, 25]
y in [0, 25]
s_1 in [-5, 5]

Applied Contractions:	Applied Constraints:
c_1.x	x <= 24 && x > 0
c_2.y	y <= 24 && y > 0
c_3.s_1	s_1 <= 2 && s_1 >= -2

Split Dimension: s_1

Reasons: c_1, c_2

s_1 <= 0

s_1 > 0

Reasons: c_2, c_3

ICP State 2

Variable Bounds:

x in (0, 24]
y in (0, 24]
s_1 in [-2, 0]

Applied Contractions:	Applied Constraints:
c_1.s_1	s_1 < 0
c_2.x	x <= 5 && x > 1
c_1.s_1	s_1 < 0 && s_1 > 0

UNSAT
Reasons: c_1, c_2

ICP State 3

Variable Bounds:

x in (0, 24]
y in (0, 24]
s_1 in (0, 2]

Applied Contractions: Applied Constraints:

Split Dimension: s_1

Reasons: c_2,c_3

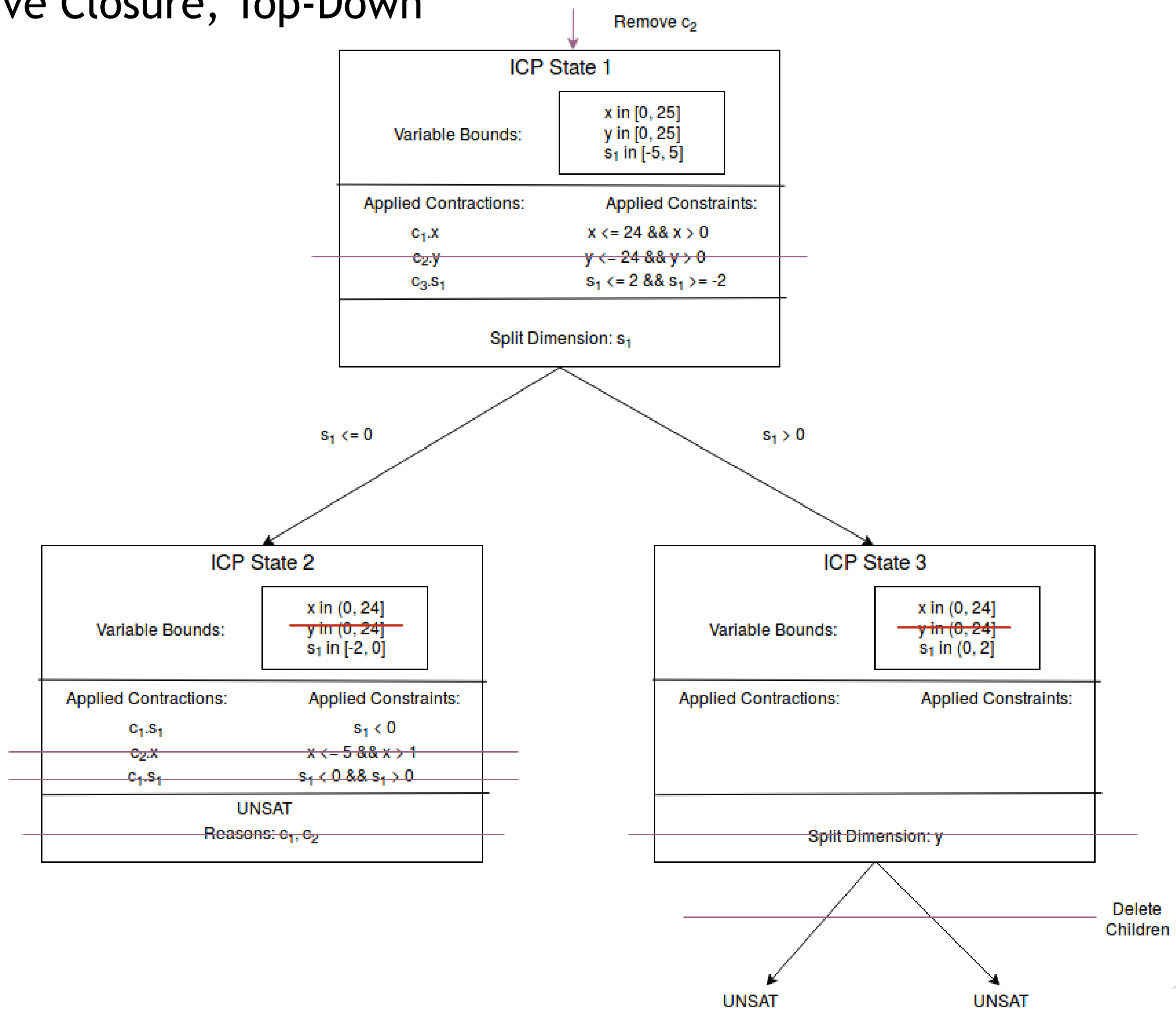
Reasons: c_3

UNSAT

UNSAT

Removing Constraints

- Transitive Closure, Top-Down



Remove c_2

ICP State 1

Variable Bounds:

x in [0, 25]
y in [0, 25]
s_1 in [-5, 5]

Applied Contractions: Applied Constraints:

c_1.x x <= 24 && x > 0

~~c_2.y y <= 24 && y > 0~~

c_3.s_1 s_1 <= 2 && s_1 >= -2

Split Dimension: s_1

s_1 <= 0

s_1 > 0

ICP State 2

Variable Bounds:

x in (0, 24]
~~y in (0, 24]~~
s_1 in [-2, 0]

Applied Contractions: Applied Constraints:

c_1.s_1 s_1 < 0

~~c_2.x x <= 5 && x > 1~~

~~c_1.s_1 s_1 < 0 && s_1 > 0~~

UNSAT

~~Reasons: c_1, c_2~~

ICP State 3

Variable Bounds:

x in (0, 24]
~~y in (0, 24]~~
s_1 in (0, 2]

Applied Contractions: Applied Constraints:

~~Split Dimension: y~~

Delete
Children

UNSAT

UNSAT

Governing Algorithm

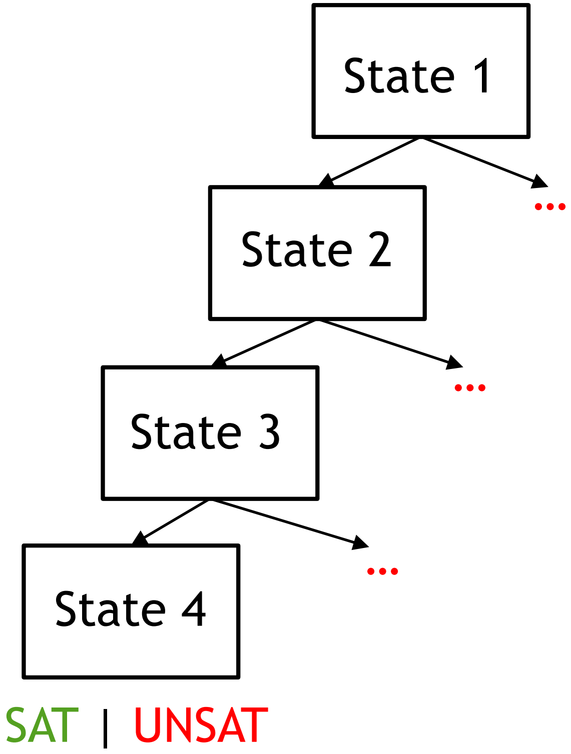
- Two priority queues: ICP States & Contraction Candidates
- Take „best“ ICP State, contract it until:
    - SAT: Return Model
    - UNSAT: Continue with next State
    - Split Occurred: Add child states to queue
    - Other Termination Conditions: Ask Backend
- If queue empty: Return UNSAT

Heuristics

1. Contractions are fast
2. Splits: Smaller Space vs. more trees

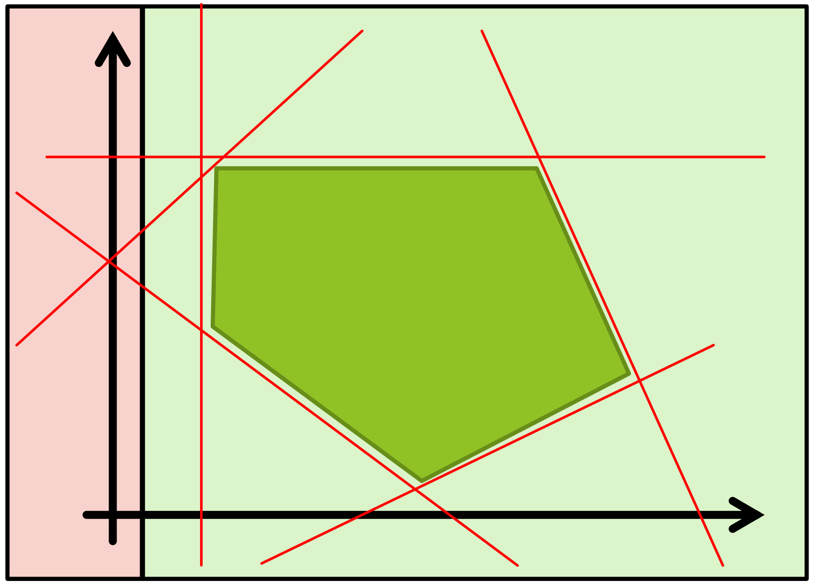
   ⇒ Contract before Splitting

- SAT vs UNSAT
  - Found SAT ⇒ problem solved, disregard all sub-trees
  - Found UNSAT ⇒ problem not solved, regard all sub-trees



Choosing next Sub-Tree

- Given several interval maps
- Each has guessed solution
- Choose the one with most fulfilled constraints
- Then the one with the smallest diameters



Choosing Contraction Candidates

- Weights/ History
- Penalize Splits
- Regard limited set of candidates:
  - 1. look at first **n** candidates in priority queue
  - 2. calculate and update weights
  - 3. choose best
  - 4. Threshold not passed?
    - Continue searching
    - Split later

Choosing Splitting Variable

- Guess solution
- Only split if variable occurs in UNSAT constraint
- Choose variable with greatest diameter
  - Most gain from splitting

Problem of Parametrization

- How to select parameter (alpha, threshold,…)?
- Available approaches: Machine Learning vs. "Empirical Testing"
- Machine Learning
  - Too complex, analysis of each problem instance required
  - Correlation/Causality between parameters
  - Deep Knowledge of the domain required

Empirical Testing

- Test for each parameter and each possible assignment
  - Too many tests, for 8 parameters, 10 possible values -> 10^8 tests (per instance)
  - for 10.000 problems -> 10.000*10^8 tests

- Abstraction Nr. 1: Create classes of problems
  - For each class: test all possible assignments, e.g., 10 classes
  - 10*10^8, still too many

- Abstraction Nr. 2: Test iteratively
  1. Select one parameter, test several assignments, select best
  2. Assign selected as new, precede with remaining attributes
  - 10*10*8 tests required (vs. 10^9)

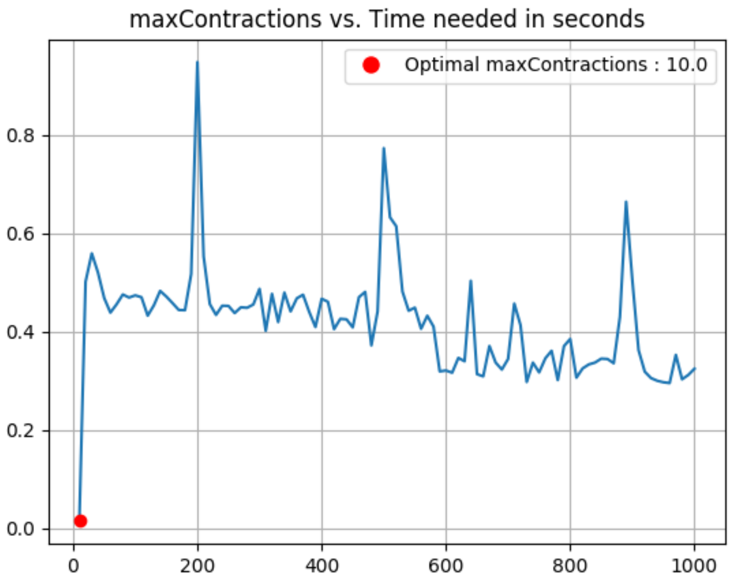
Infrastructure

- Adapted Parametrization:
  - Parametrization at run-time (and not compile-time)
  - New Class: Dynamic Settings (adapt settings from file)
  - Simple Python infrastructure for generation, processing and visualization of parameters

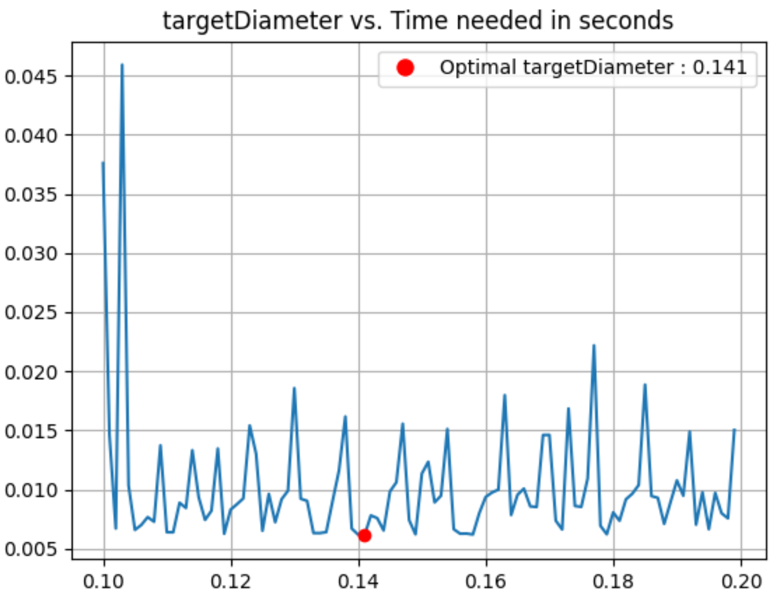
- Problem left: Classification
  - Too many classes -> Testing not possible
  - Too few classes -> Testing too schematic

Results (Small Problem)

- Tested : meti-tarski/sin/cos… classified by required computation time
- Possible classifications: problem size, degree of polynomials, etc.



maxContractions vs. Time needed in seconds



targetDiameter vs. Time needed in seconds