Evaluating Branching Heuristics in Interval Constraint Propagation for Satisfiability

Calvin Huang	Scale Labs
Soonho Kong	Toyota Research
Sicun Gao	UCSD
Damien Zufferey	MPI-SWS

NSV 2019
13.07.2019

Branch and Prune ICP

- Used to numerically solve systems of non-linear equations

$$\exists^{[3.0,3.14]} x_1 . \exists^{[-7.0,5.0]} x_2 . 2\times 3.14 - 2 x_1 \arcsin\left(\cos 0.8 \times \sin\left(\frac{3.14}{x_1}\right)\right) \leq -0.6 - 0.03 x_2 + 1.5$$

- Interval Constraint Propagation (ICP)

  Hypercube over-approximation of the solution space of non-linear constraints.

- Efficient (numerical) but incomplete (finite precision)

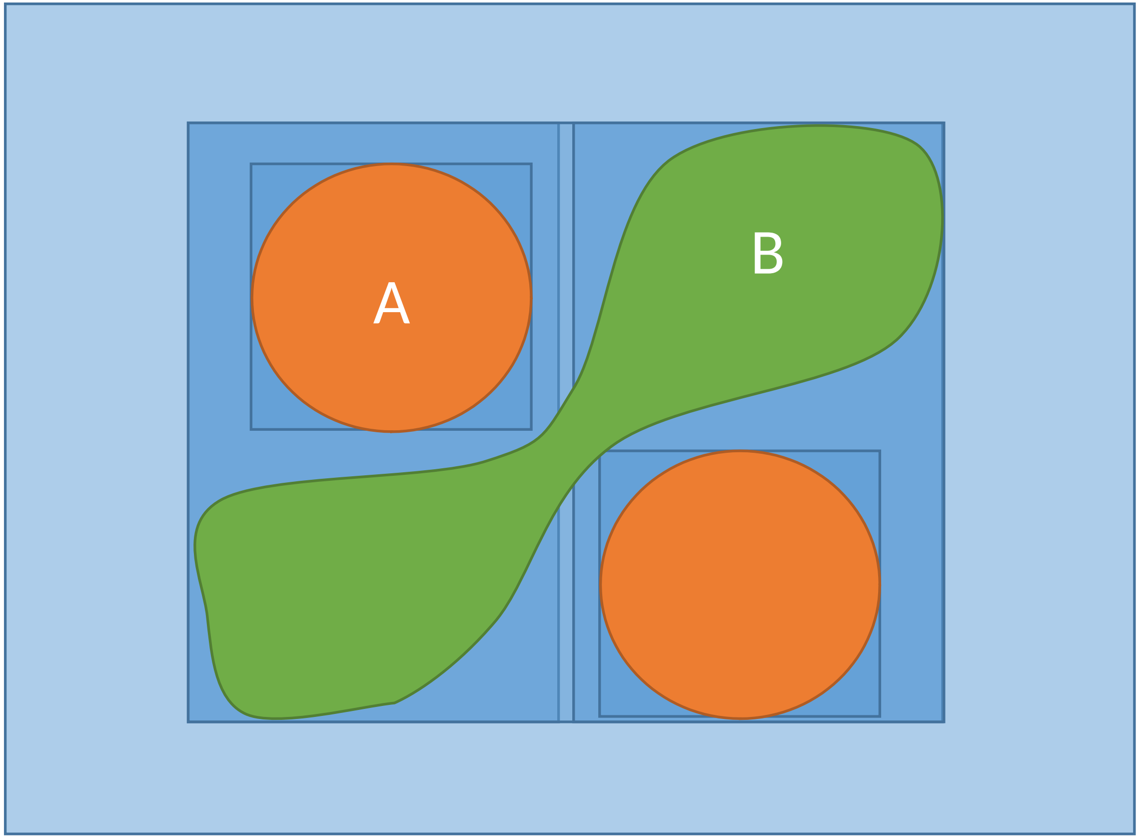
Branch and Prune Algorithm

$\text{ICP}(c_1, ..., c_m, \boldsymbol{D} = D_1 \times \cdots \times D_n, \delta)$	Variables have a bounded domain

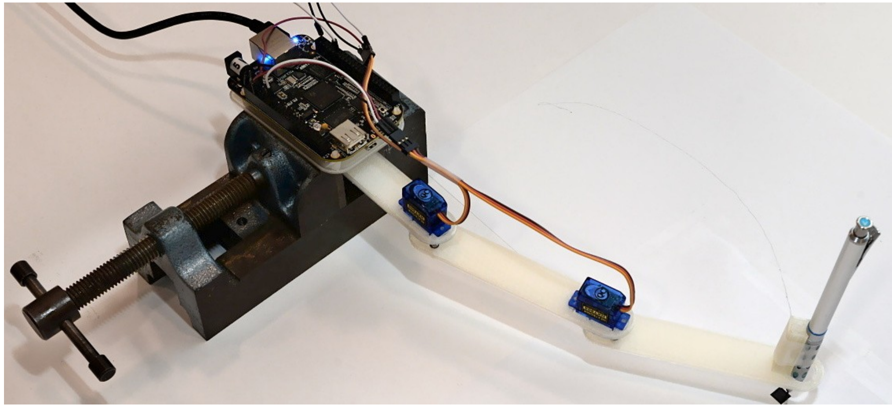
$S.\text{push}(\boldsymbol{D})$
while $S \neq \emptyset$ **do**
  $\boldsymbol{D} \leftarrow S.\text{pop}()$
  while $\exists 1 \leq i \leq m, \boldsymbol{D} \neq_\delta \text{Prune}(\boldsymbol{D}, c_i)$ **do** — Pruning: constraint propagation
    $\boldsymbol{D} \leftarrow \text{Prune}(\boldsymbol{D}, c_i)$
  end while
  if $\boldsymbol{D} \neq \emptyset$ **then**
    if $\exists 1 \leq i \leq n, |D_i| \geq \delta$ **then** — Terminate: reached a given precision
      $\{\boldsymbol{D}_1, \boldsymbol{D}_2\} \leftarrow \text{Branch}(\boldsymbol{D}, i)$ — Branch: divide and conquer
      $S.\text{push}(\boldsymbol{D}_1)$
      $S.\text{push}(\boldsymbol{D}_2)$
    else
      return sat
    end if
  end if
end while
return unsat — Terminate: no possible solution

Branch-and-Prune ICP Algorithm

Prune by B
Prune by A
Branch
Prune by A
Prune by B
Prune by A
Prune by B



Then Came Some New Benchmarks



Small robotic arm partly designed and programmed using constraint solving.

- Generated constraints
  - Each part
    - 7 variables for the pose
    - variables for dimensions
  - Joints are constraints
  - Planning by unfolding the constrains
  - >100 var, >500 constraints

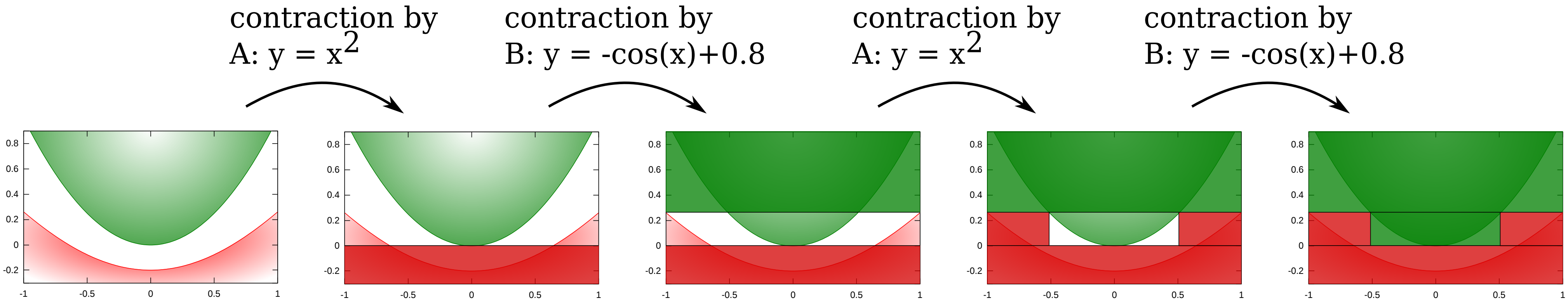
Scaling ICP, Paving, and SAT

- The search is exponential in the number of dimensions.

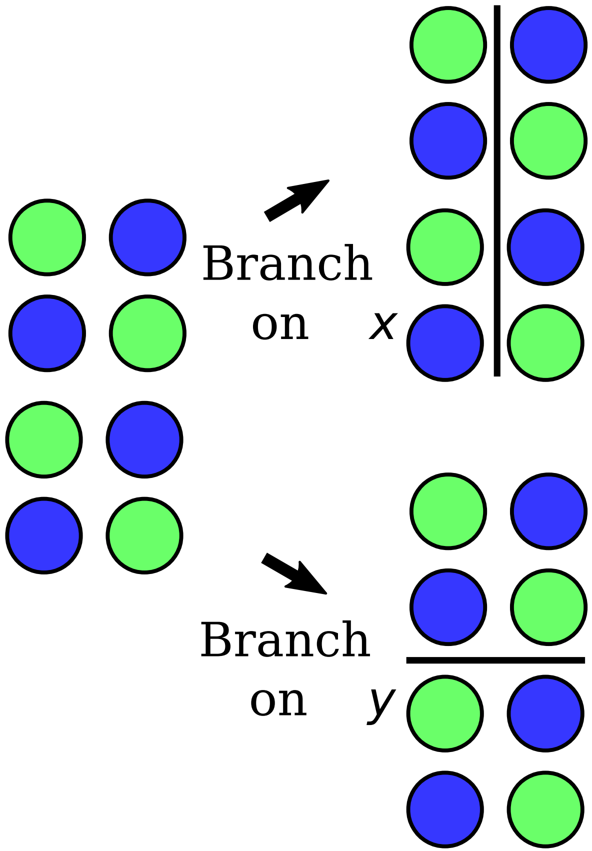
  Which variable is split can make a huge difference.
- Uses for ICP
  - Paving: map the *entire* solution space
  - Satisfiability: stop when *one* solution is found
- Evaluations in the literature focus on paving, we wanted to see what happens with satisfiability.

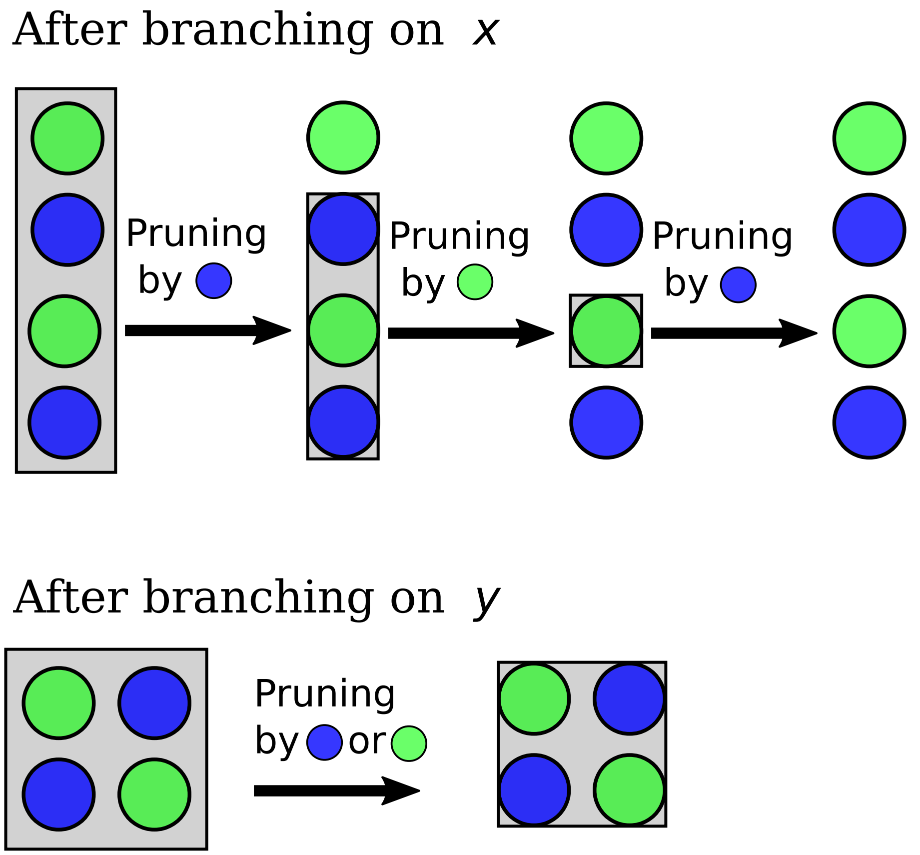
Finding All Solutions

Turns complicated non-linear constraints into a collection of hypercubes.

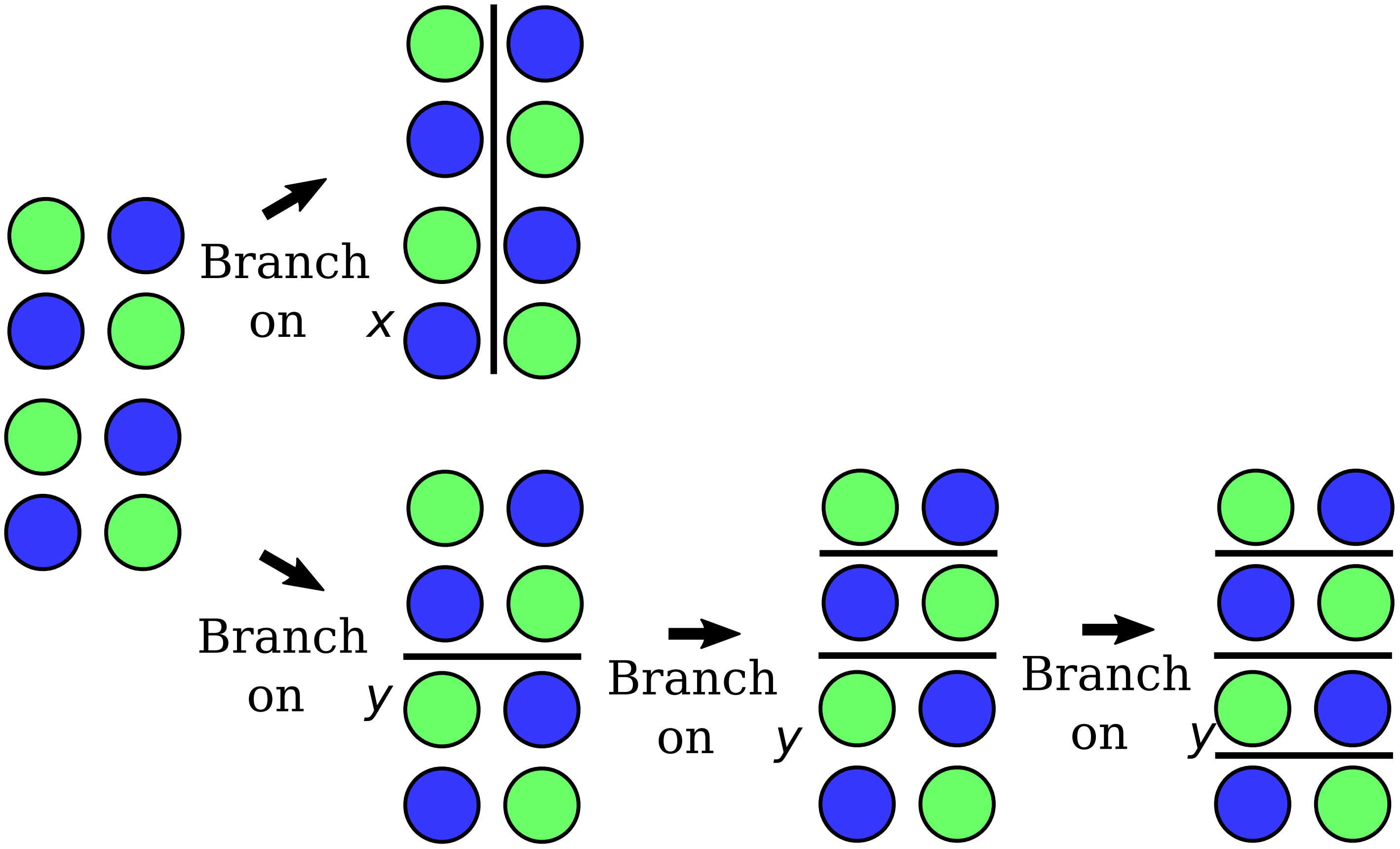


Branching Matters (1)





Branching Matters (2)

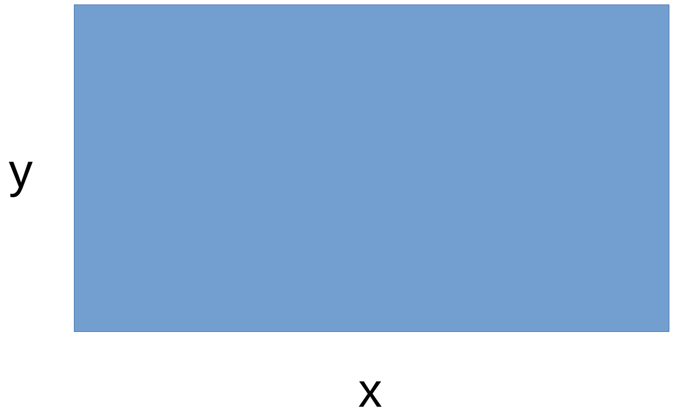


Branching Heuristics

- Compare three heuristics:
  - Largest First
  - Smearing
  - Lookahead
- Try to isolate the effect of each heuristic:
  - No combinations of methods (no kitchen sink approach)
  - Each method has parameters, we pick some value on a few examples and then did not change it (no over-fitting)

Largest First (Baseline)

- Pick the variable with the largest range.
- + simple
- - agnostic to the constraints or search history



Choose x

Smearing (Gradient)

- Evaluate the Jacobian at the center point.
- Rank variables by

  sum of partial derivatives multiplied variable domain

- Example: $\begin{array}{c} x+5y=0 \\ x\in[0;10] \\ y\in[0;3] \end{array}$, branch on y (x score is 10, y is 15)

- + exploits information about the constraints
- - more complicated the largest first

Lookahead

- Split along all the dimensions and do one step of pruning.
- Keep the choice that worked best.

- Idea: “wrong” split doubles the search, find the “best” choice

- + locally optimal (does not translate to globally optimal)
- - can be very expensive

Early Hopes

- Excerpt from an email exchange with Calvin:

  ... I implemented the gradient splitting, and it's better sometimes, worse sometimes. ...

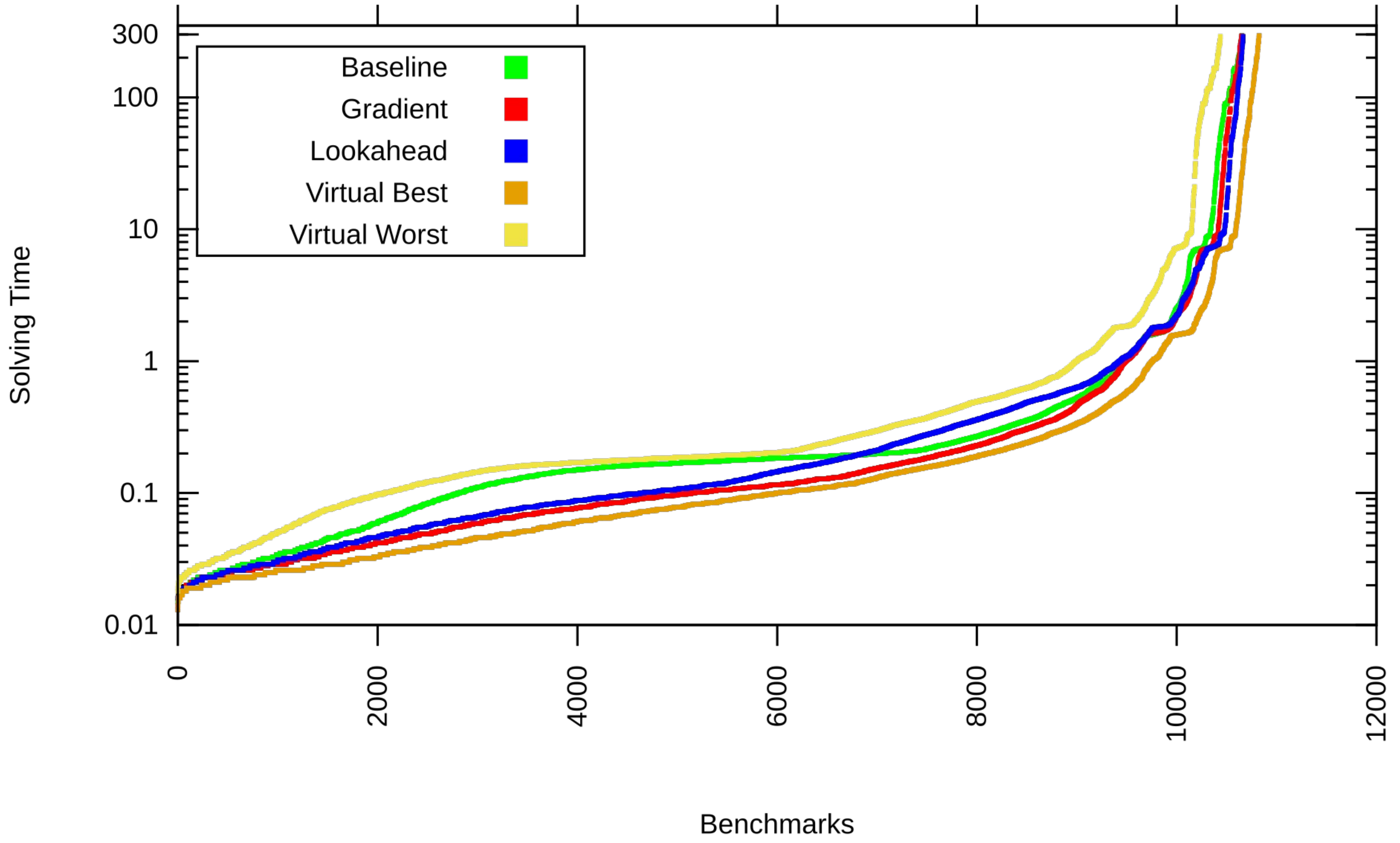
Test Name	Largest First	Smearing
mass_spring	**0.098**	0.336
model0a	12.109	**0.016**
model2	**0.185**	TO
oneParam	**3.148**	7.344

Test Name	Largest First	Smearing
12	**0.154**	0.176
simdreal_4	1.729	**0.206**
simdreal_5	2.259	**0.173**
stephen_01	0.212	**0.014**

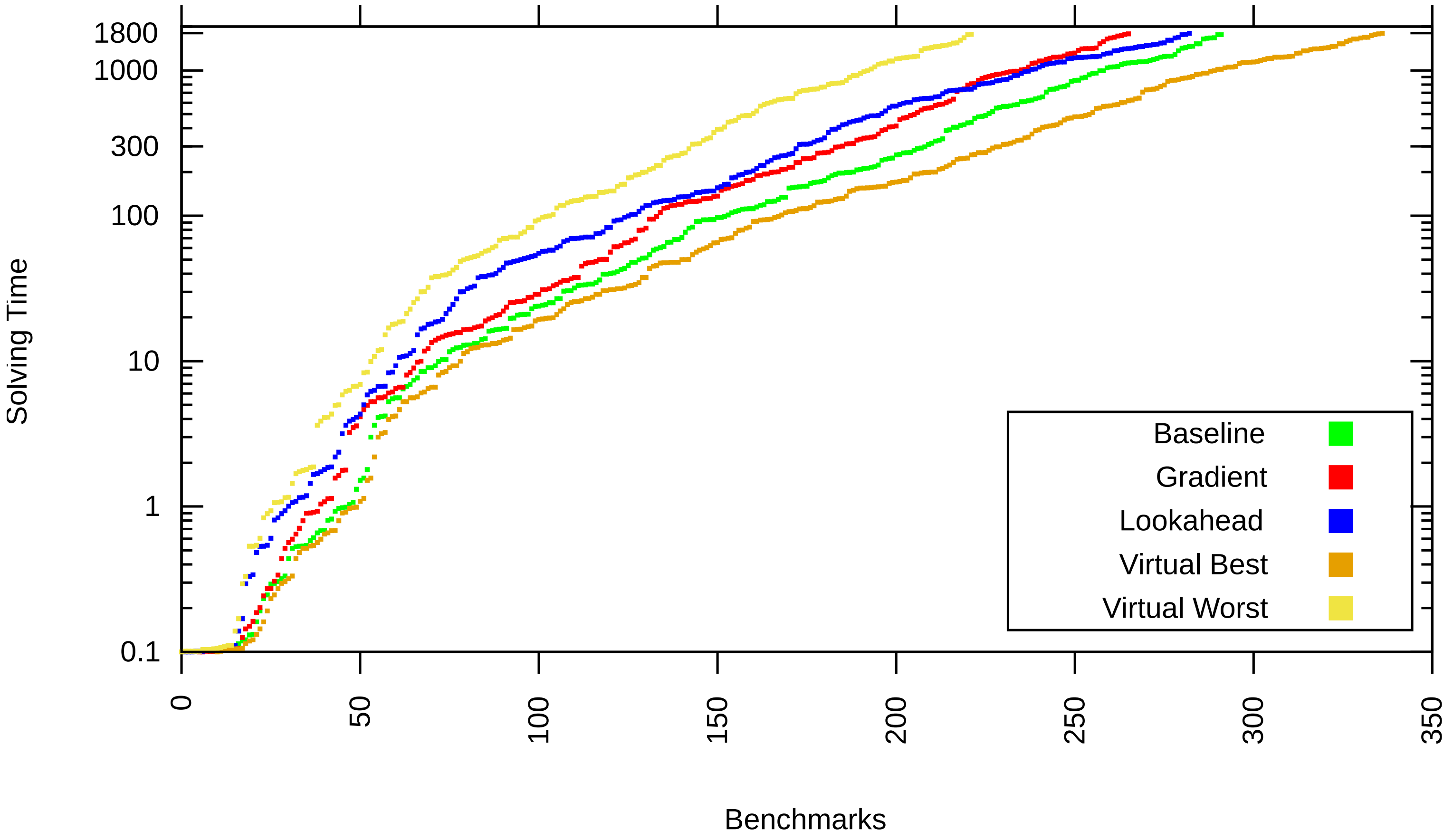
Benchmarks and Tests

- Tested with the dReal SMT solver for QF_NRA.
- It is easy to fall in the trap of over-fitting heuristics.
  - We need data: over 11,000 benchmarks
- What is a representative set of benchmarks?
  - SMT-Lib, Flyspeck, robotics, control, information th, ...
- We split the benchmarks in two categories:
  - Small: all tests, timeout of 300 sec.
  - Large: at least 8 real variables, 1800 sec. (896 tests)
- Available at https://github.com/dreal/benchmarks

Results: Small Instances



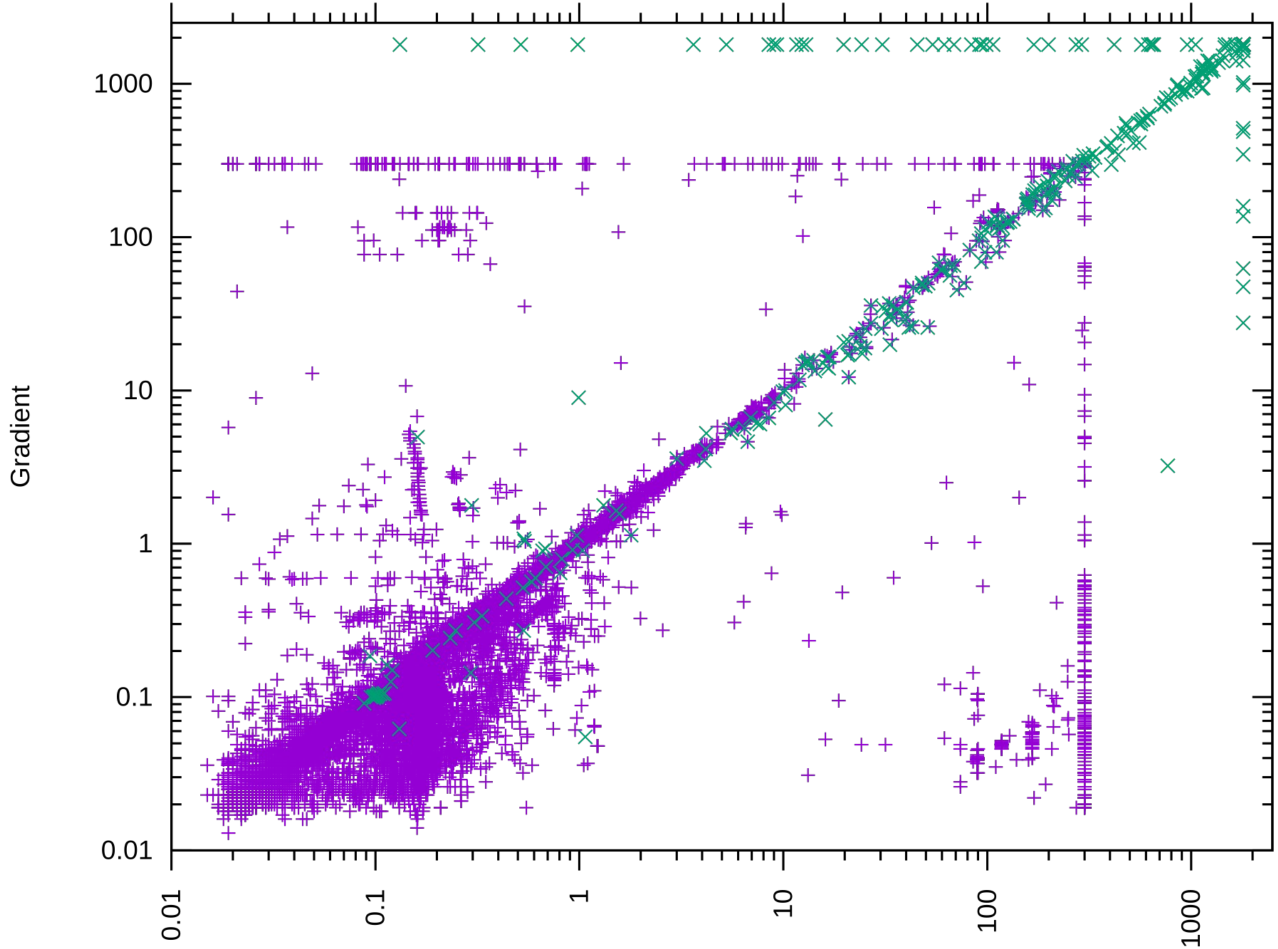
Results: Large Instances



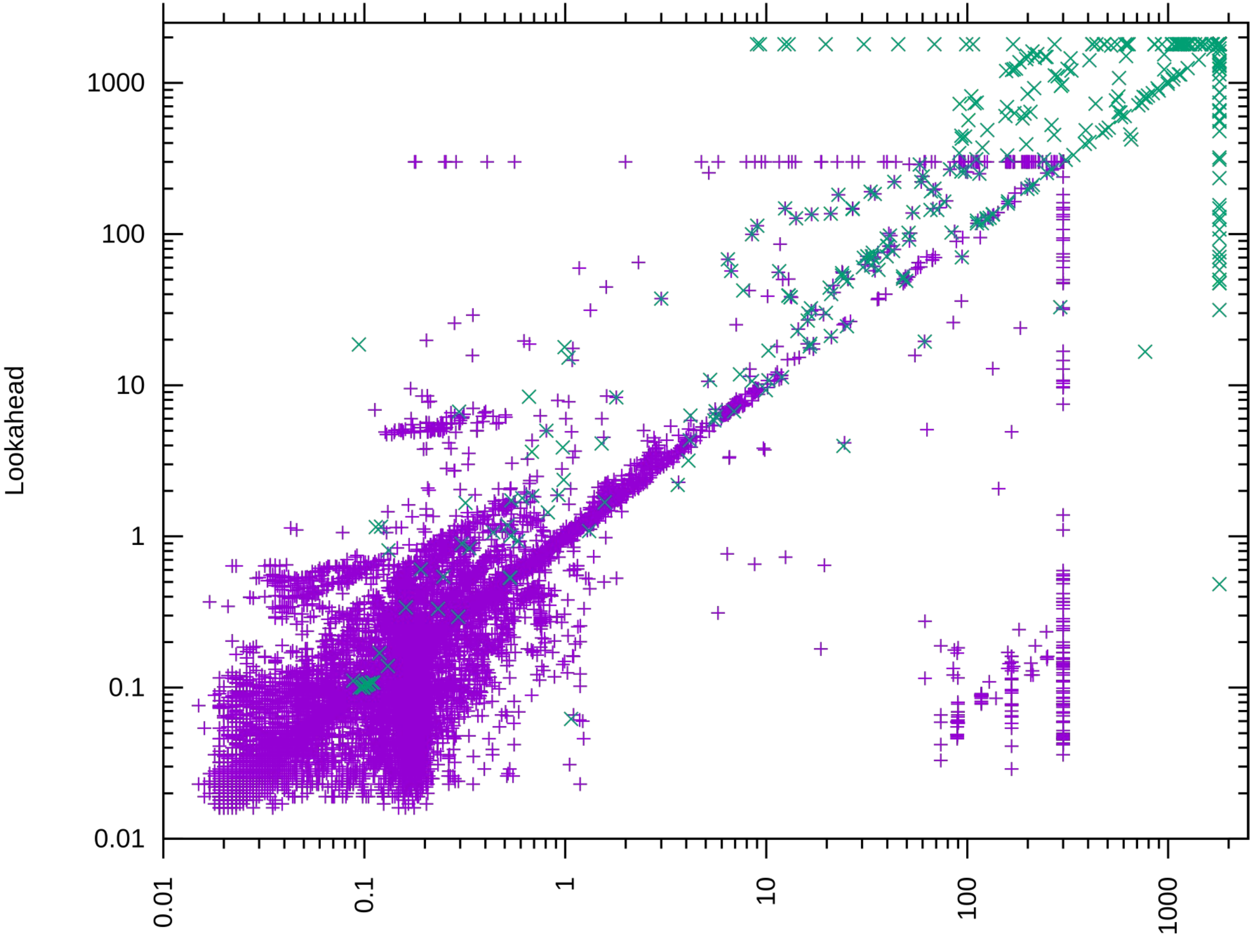
No Clear Winner

Instances	Small (Δ)	Large (Δ)
#Benchmarks	11789	896
Solved Baseline	10654	**292**
Solved Gradient	10654 (+0)	266 (-26)
Solved Lookahead	**10667** (+13)	283 (-9)
Virtual Best	10827 (+173)	337 (+45)
Virtual Worst	10439 (-206)	222 (-70)
Unique Baseline	34	19
Unique Gradient	**65**	5
Unique Lookahead	19	**31**

Smearing vs Largest First

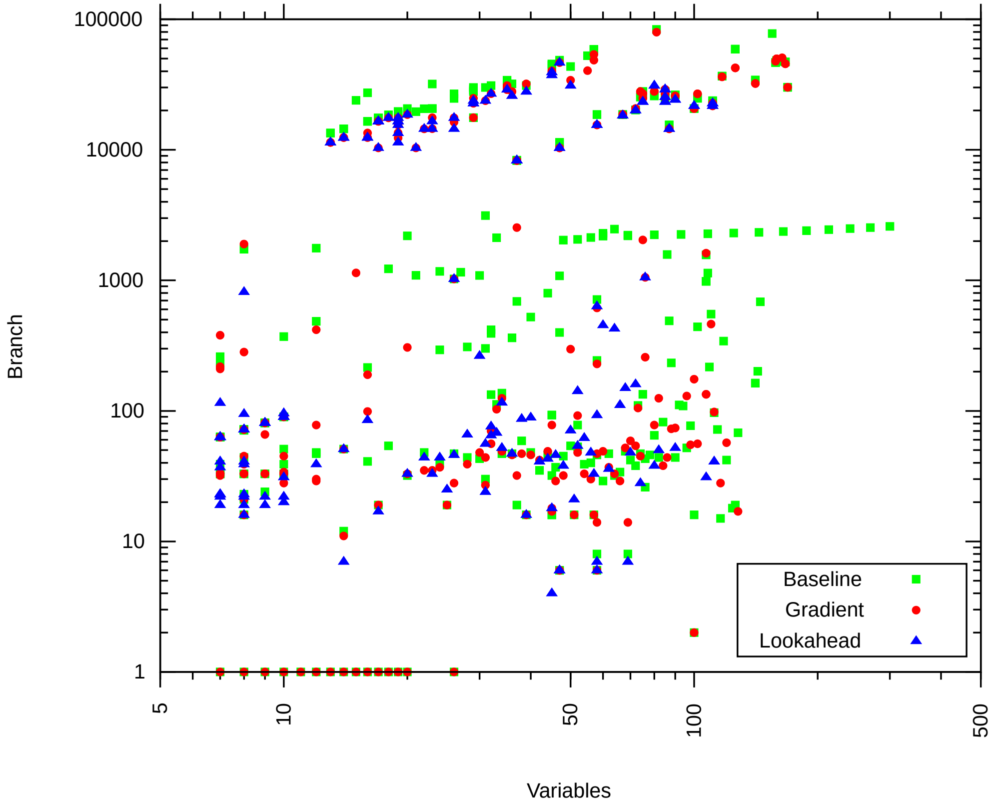


Lookahead vs Largest First

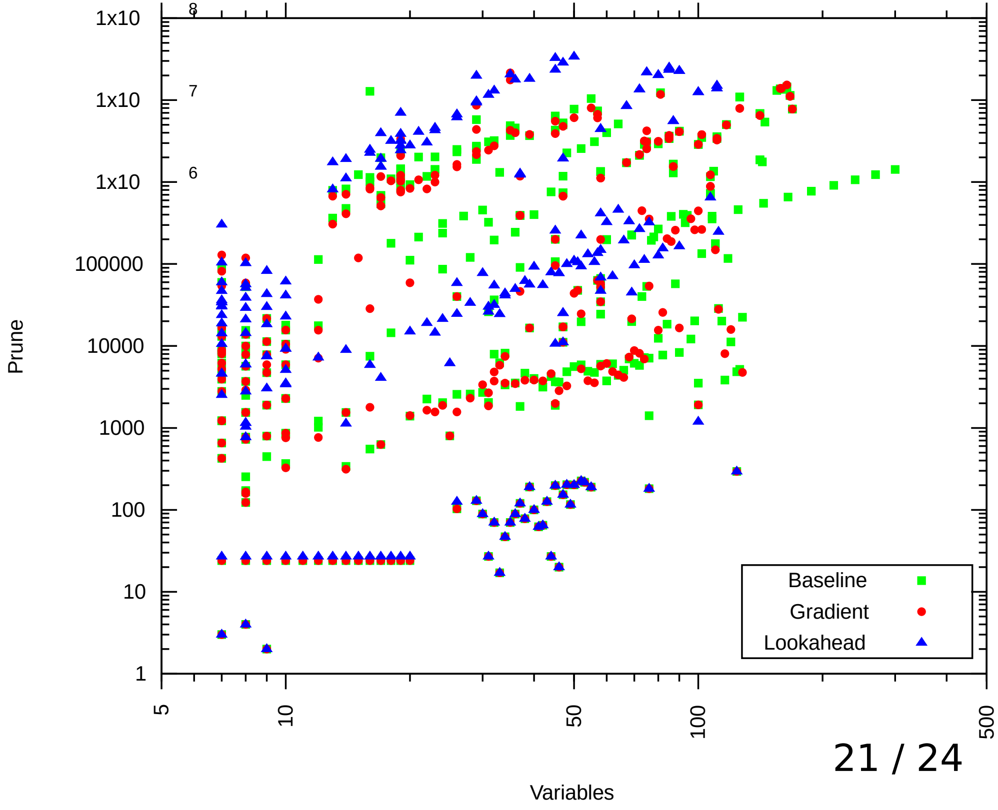


Branching is Key (Large Instances)

branching steps



pruning steps



Where Do We Go Now?

- Portfolio
  - It is an NP problem after all.
- Concurrency: New version of dReal is parallel
  - Easy parallelism across the branches of the search
  - parallel lookahead and synchronization overhead ?
- Gathering an even wider variety of benchmarks

Largest First is Easy to Trick

- Back to our robotic example. We needed solutions…
- Identify important variables (domain specific knowledge)
- Scale the dimensions: importance ~ range
- 2 orders of magnitude speed-up

- It it not possible to use such trick with the other methods.

Take Home Message

- We evaluated three branching heuristics for ICP.

  Largest first, Smearing, Lookahead
- Large set of benchmarks made publicly available.
- Unfortunately, no conclusive results yet ...

Questions ?