Data Structures for Generalised Arc Consistency for Extensional Constraints

Ian P. Gent, Chris Jefferson,
Ian Miguel and Peter Nightingale

Finite domain constraint satisfaction problem (CSP)

- Variables with a finite domain
  - e.g. $A \in \{2, 3\}$, $B \in \{1, 2, 4\}$
- Constraints placed on variables
  - $A = B$, $A + B = 4$
- A solution is a valid assignment to all variables
  - $A = 3$, $B = 1$
- NP-complete decision problem

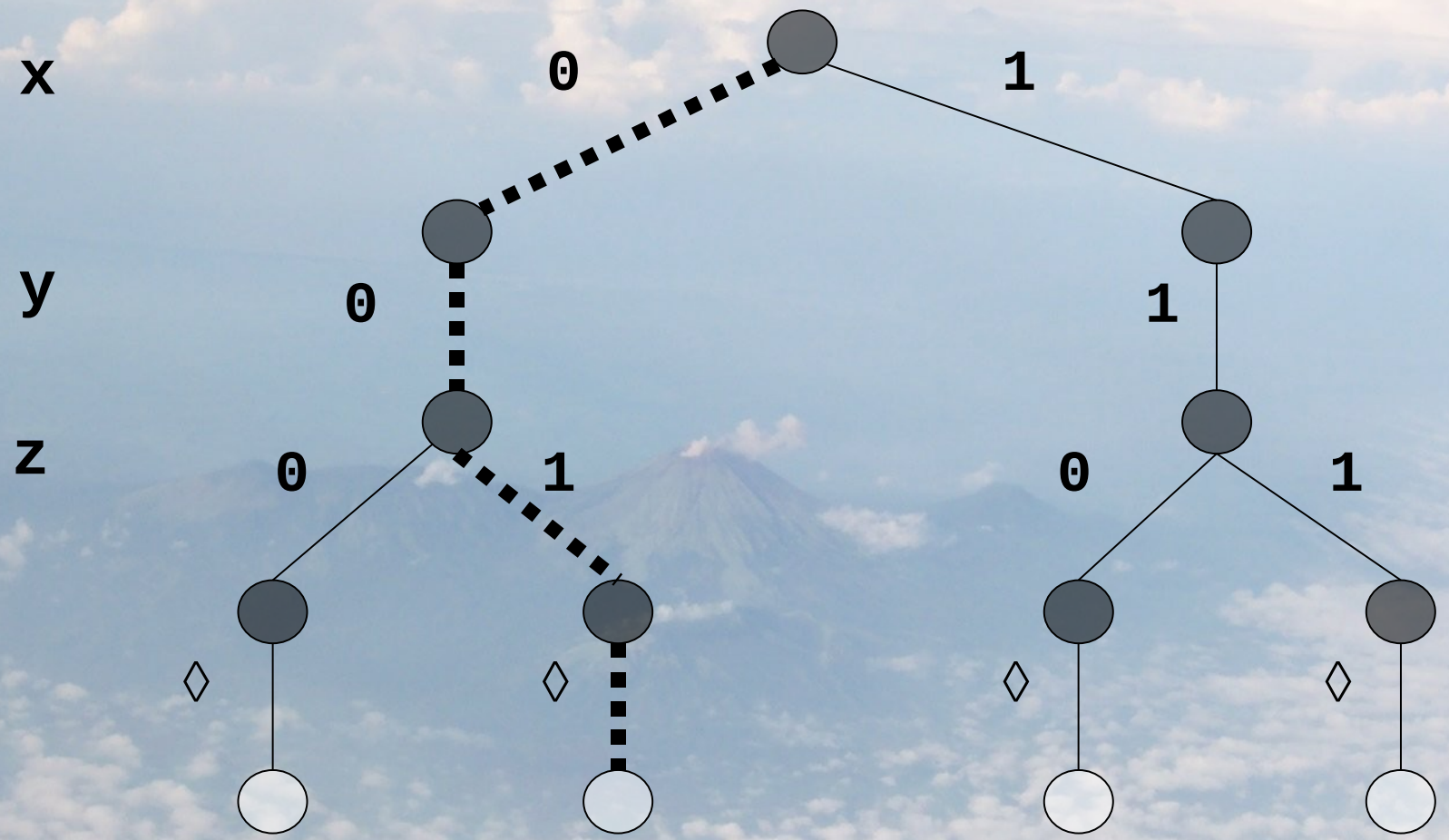
Extensional constraints

- Constraints expressed as a table of allowed combinations of values (*tuples*)
- Can express any constraint, albeit with practical limits on the number of tuples
- Useful for constraints which cannot be efficiently translated into constraints provided by the solver
  - Constraints with unusual structure
  - Used in BIBD, Graceful Graphs, Semigroup counting, Golomb ruler...

GAC

- Various algorithms to enforce GAC
  - If a value is not contained in any *valid* and *allowed* tuple, it cannot be part of any solution to the CSP instance, so remove it
  - Requires fast search through allowed tuples list for the next *valid* tuple
- We test with GAC-Schema and Minion's watched literal table constraint

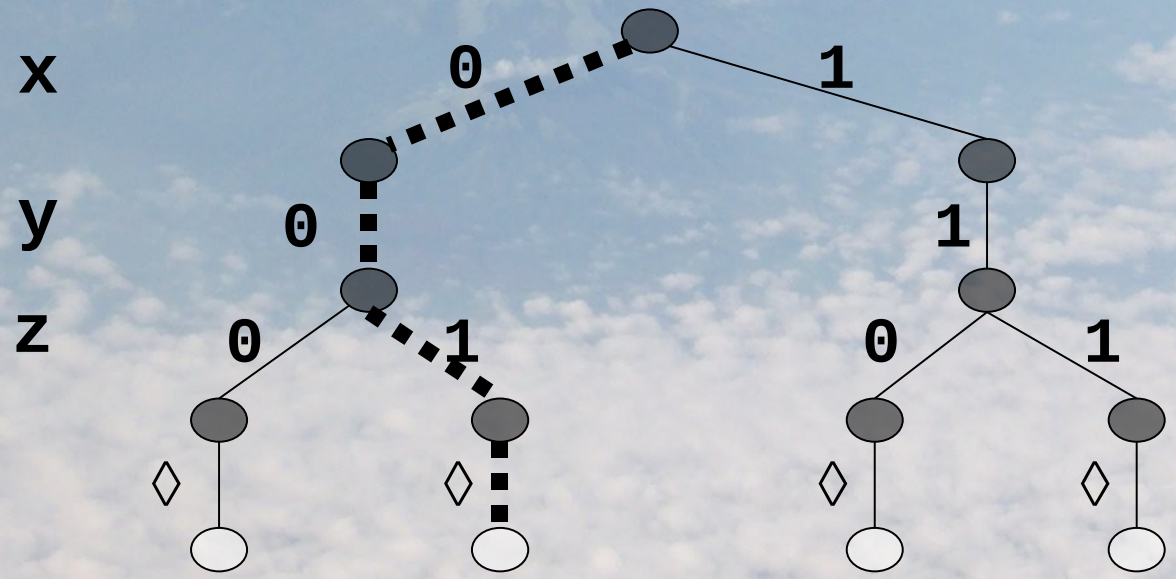
Tries



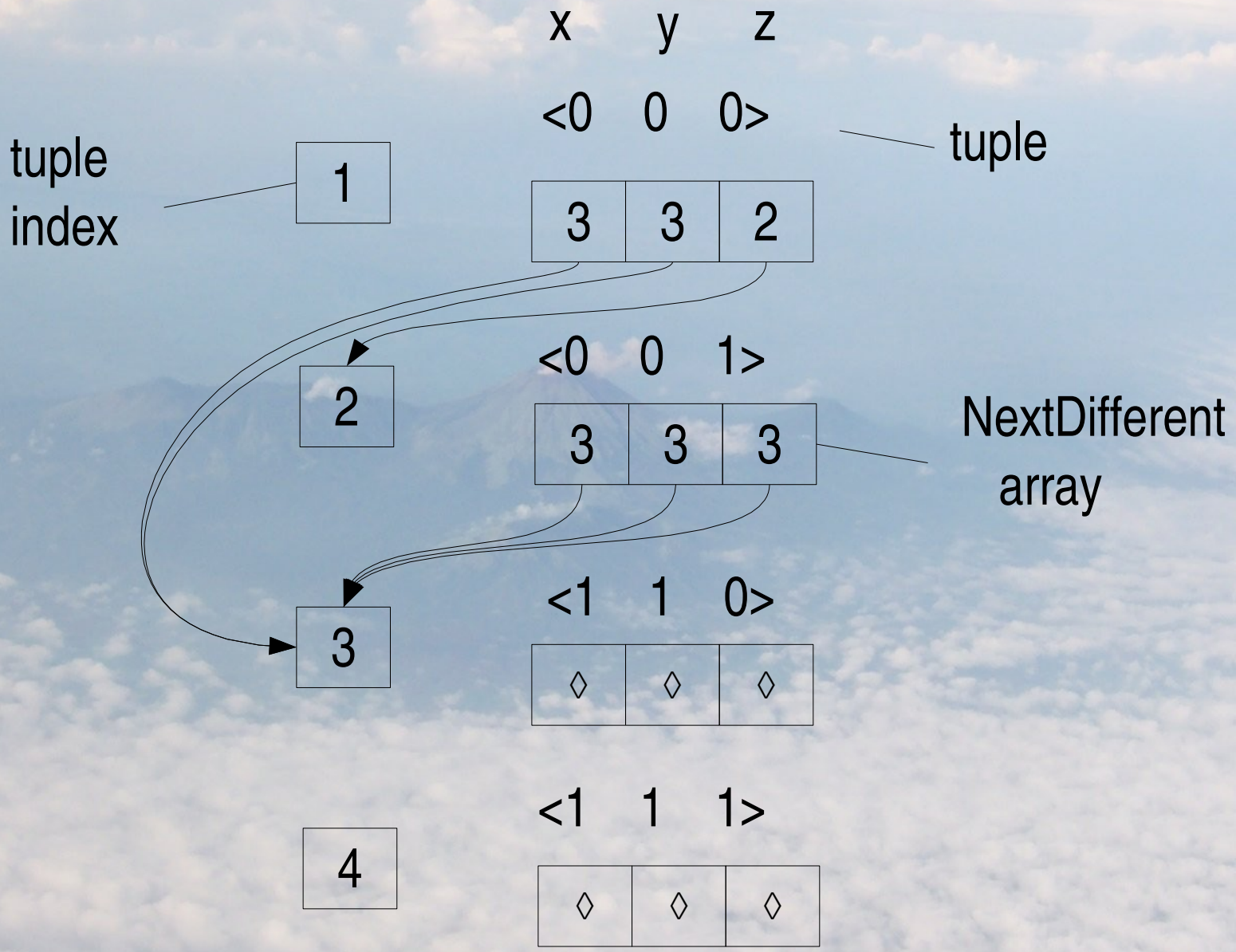
- Also have orderings y,x,z and z,x,y

Tries

- Tries are searched depth-first, following only branches for values which are in their respective domain
- To find a second tuple, search is resumed from the leaf node



Next-Difference Lists



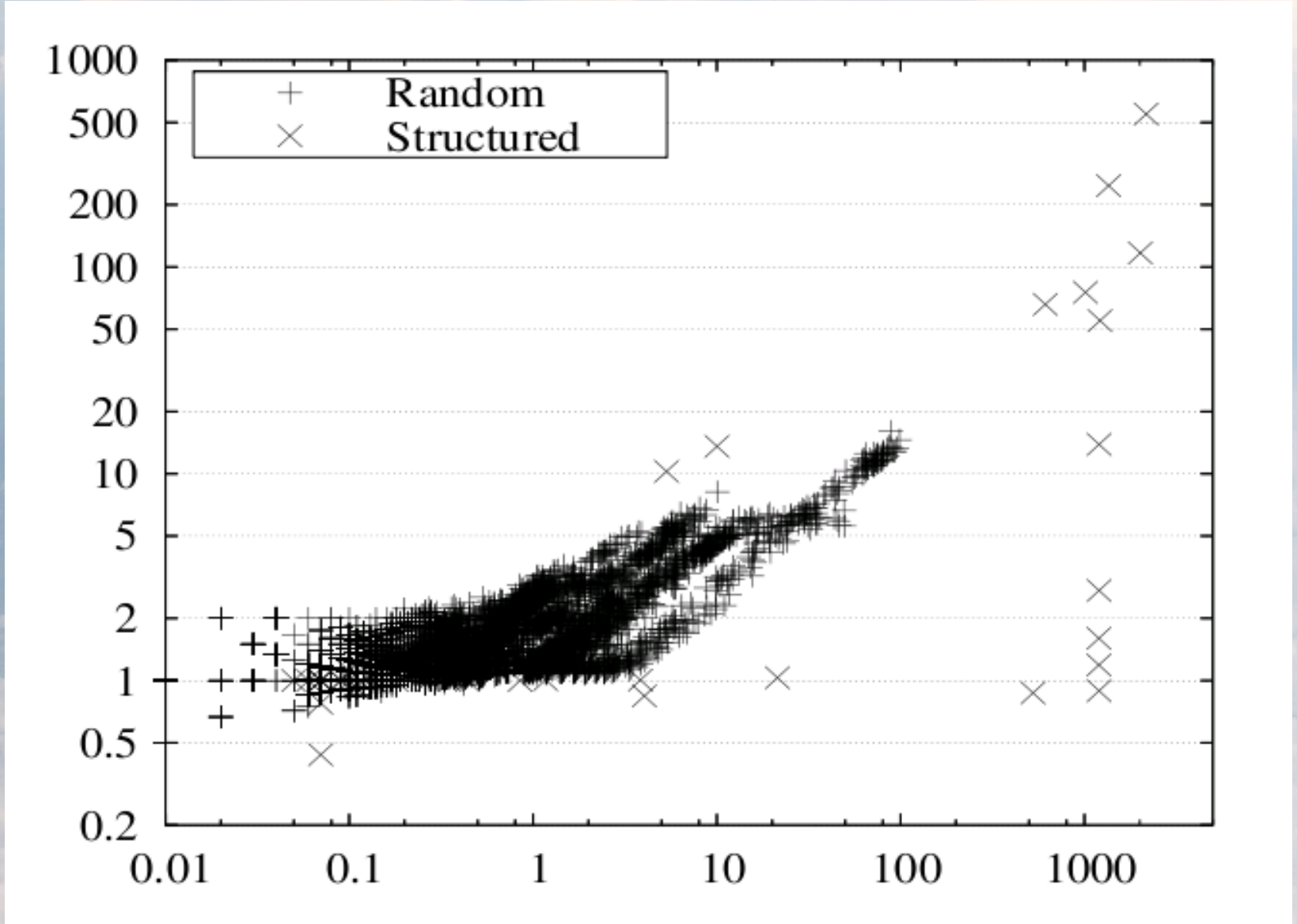
Next-Difference Lists

- Next-Difference lists sometimes able to jump forward further than tries, never less far.
- Next-Difference lists slightly more expensive
  - Iterates from beginning of tuple at each step

Comparisons

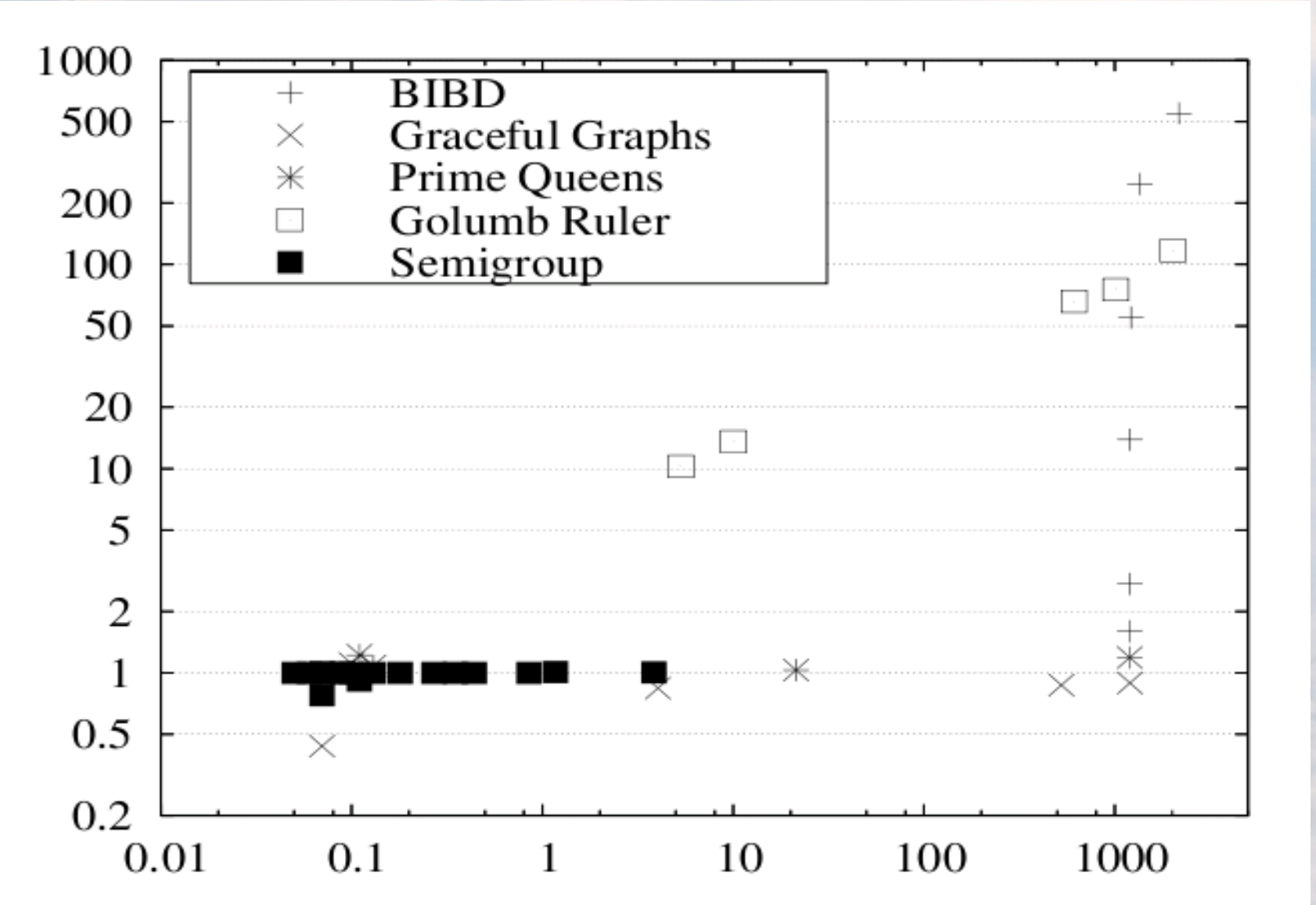
- Lecoutre and Szymanek (2006)
  - Algorithm based on binary search (Binary)
- Lhomme and Régin (2005)
  - New *Hologram* data structure (Hologram)
- Bessière and Régin (1997)
  - Algorithm which iterates through the list (Simple)
- Comparison in context of Minion's watched literal adaptation of GAC-2001

Tries vs. Simple



Time limit of 1200s

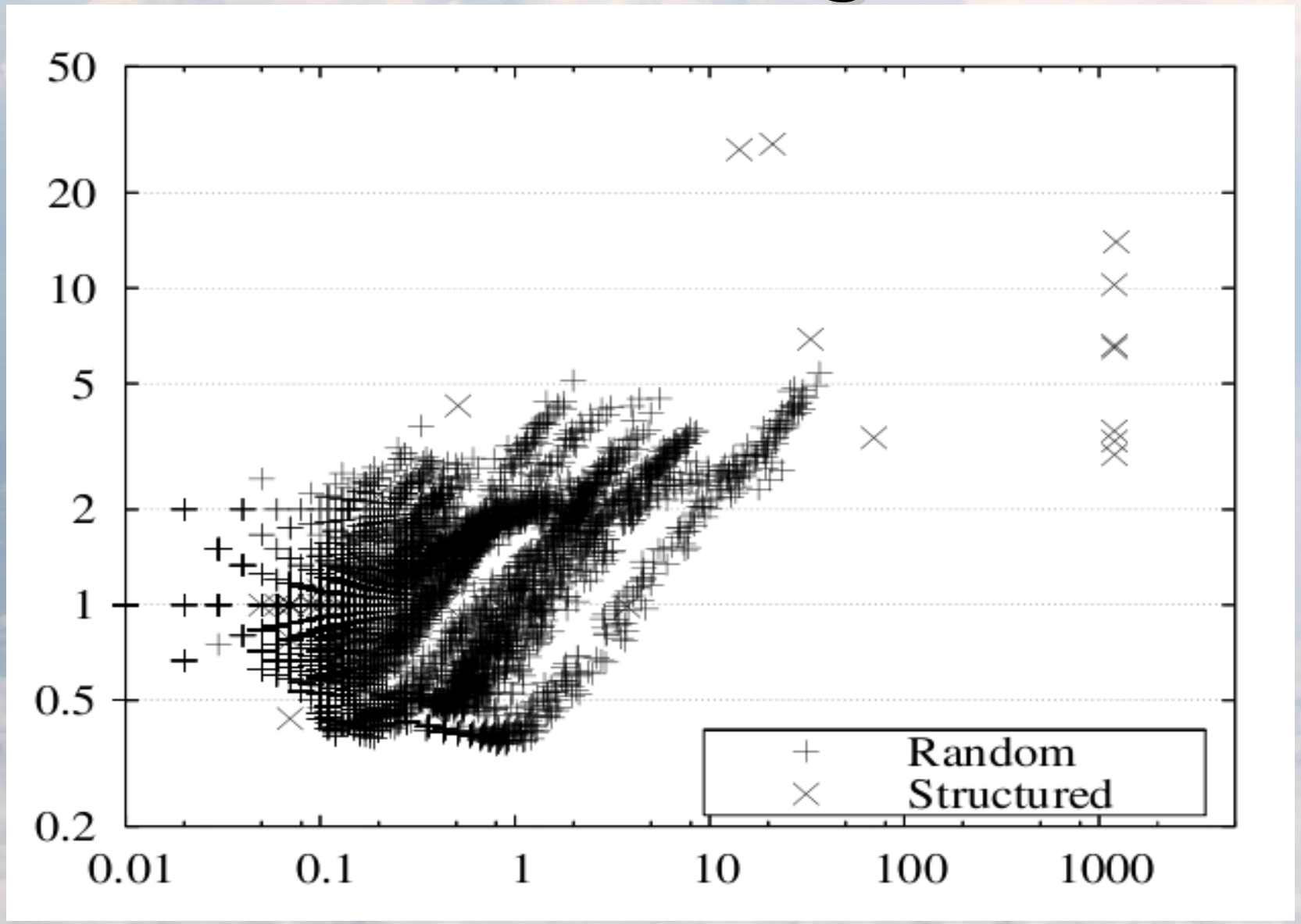
Tries vs. Simple



Nodes per second ratio

Run-time for Simple (s)

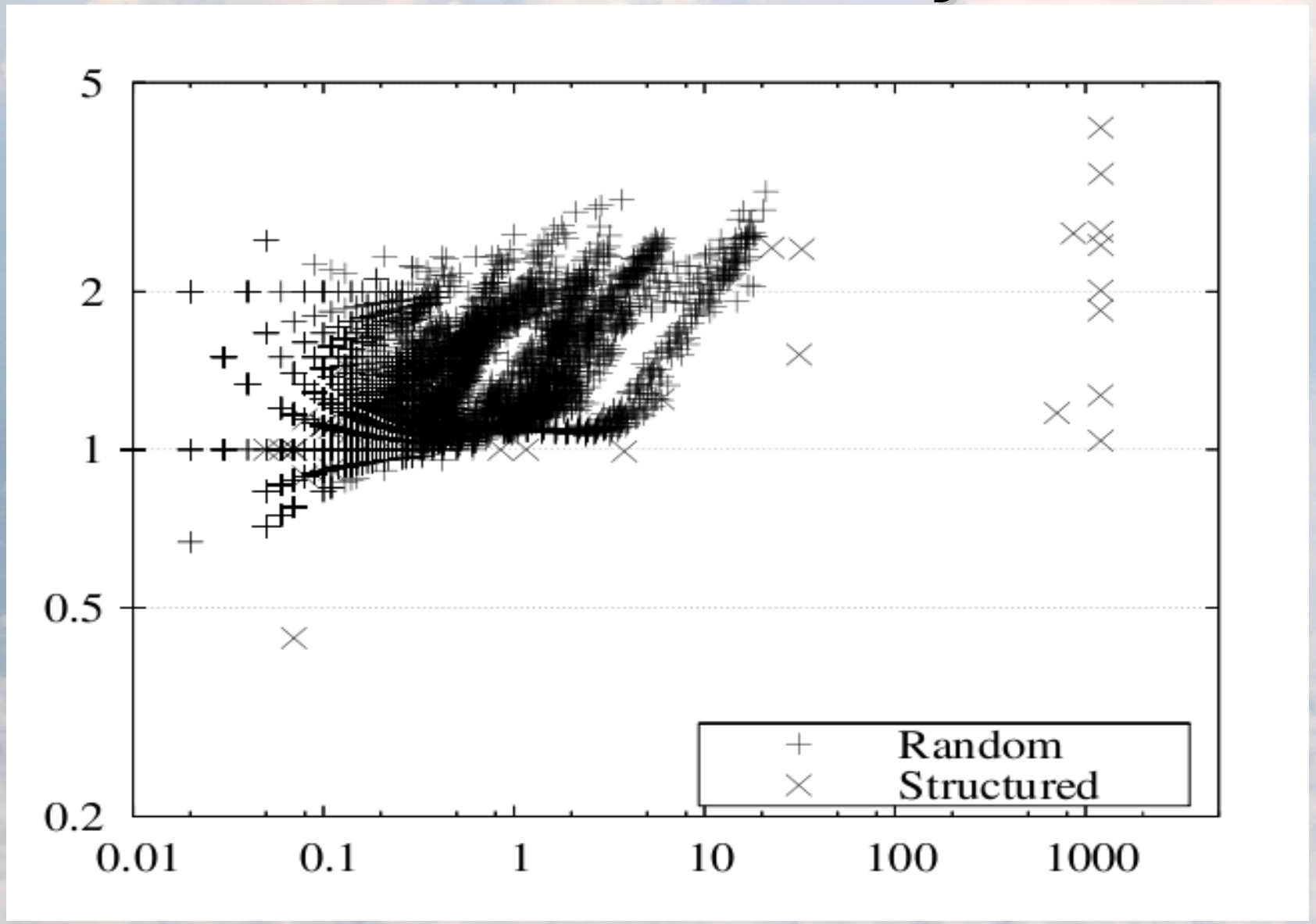
Tries vs. Hologram



Nodes per second ratio

Run time for Hologram (s)

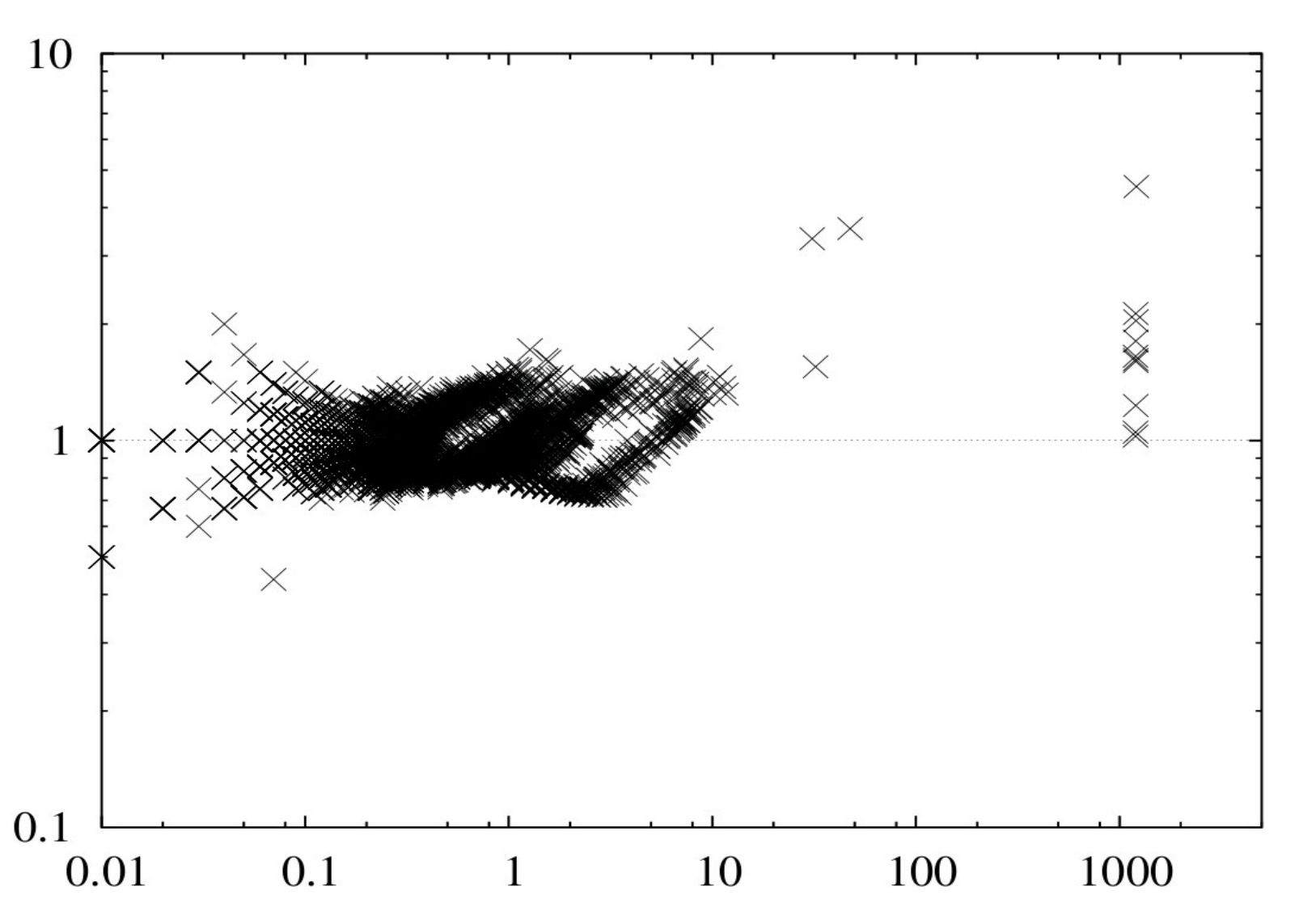
Tries vs. Binary



Nodes per second ratio

Run time for Binary (s)

Tries vs. Next-Difference Lists



Conclusions

- Proposed two new methods
  - Tries somewhat more effective
- Built empirical case that Tries scales better than Hologram or Binary
  - Both random and structured instances

Thank you

- Any questions?

Structured problems

- Graceful Graphs: ternary constraints
- Prime Queens: ternary constraints
- Golomb Ruler: quaternary and ternary table constraints