Value-Ordering and Discrepancies

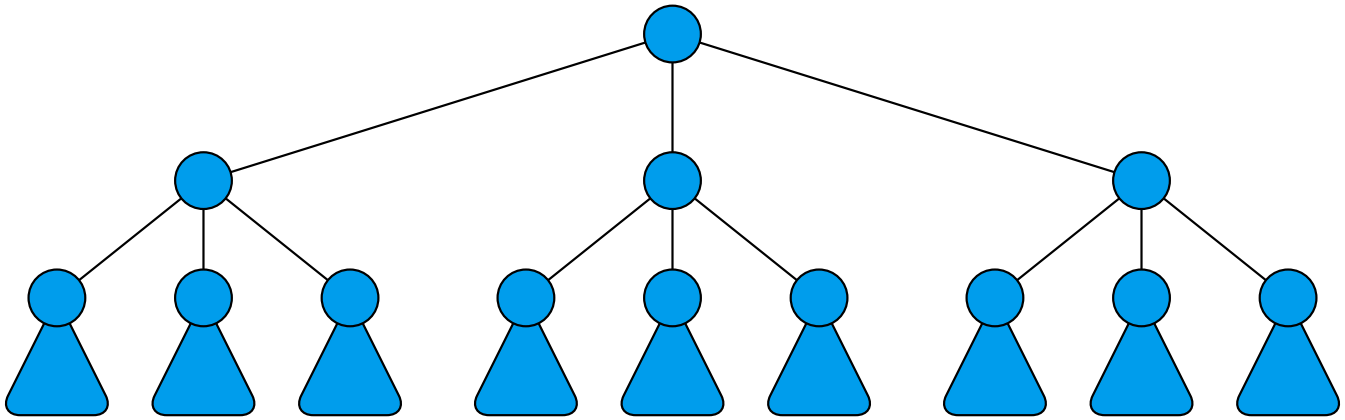
Ciaran McCreesh and Patrick Prosser

University of Glasgow

Maintaining Arc Consistency (MAC)

- Achieve (generalised) arc consistency (AC3, etc).
- If we have a domain wipeout, backtrack.
- If all domains have one value, we're done.
- Pick a variable (using a heuristic) with more than one value, then branch:
  - Try giving it one of its possible values (using a heuristic), and recurse.
  - If that failed, reject that value, pick a new value, and try again.
  - If we run out of values, backtrack.

Search as a Tree



- Circles are recursive calls, triangles are ‘big’ subproblems.
- Heuristics determine the ‘shape’ of the tree.
- MAC is like Depth-First Search (DFS).

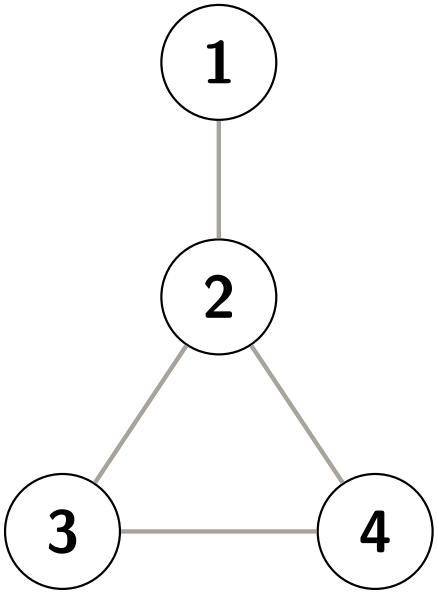
Variable-Ordering Heuristics

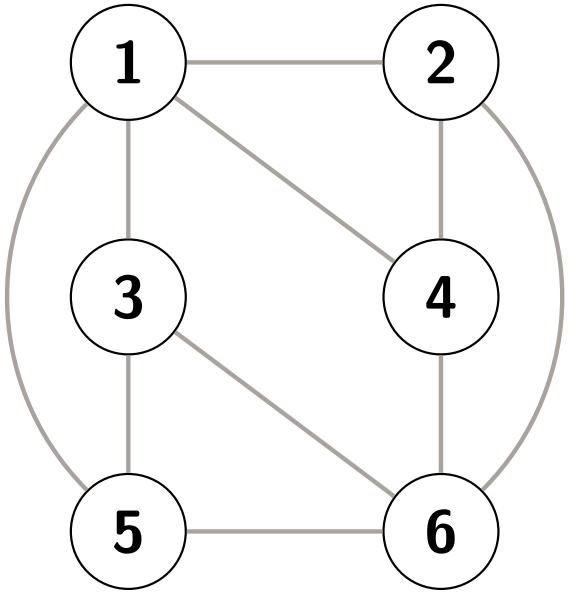
- Variable-ordering heuristics determine the number of children at each level of the search tree.
- We have quite good general-purpose variable-ordering heuristics:
  - Smallest domain first (but not for 0/1 encodings).
  - Most constrained first.

Value-Ordering Heuristics

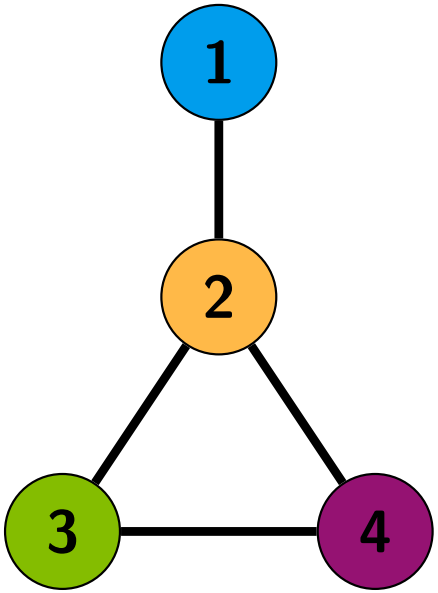
- Value-ordering heuristics determine the paths taken through the search tree.
- Designing value-ordering heuristics can be harder...
- For MAC, value-ordering heuristics only matter for satisfiable instances, and for optimisation problems.

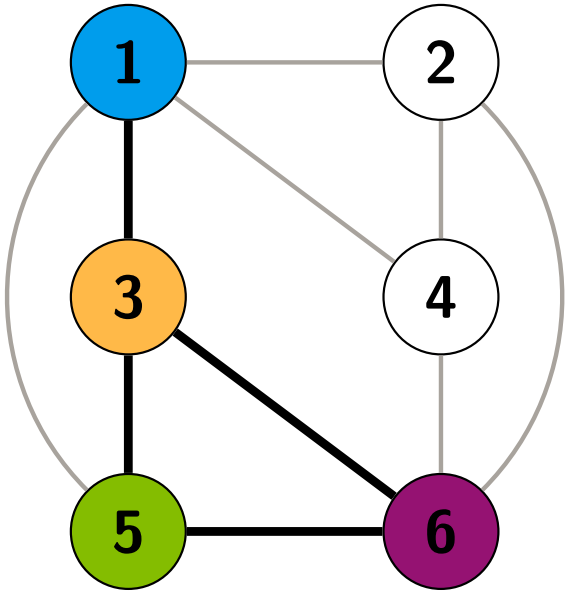
Subgraph Isomorphism





Subgraph Isomorphism

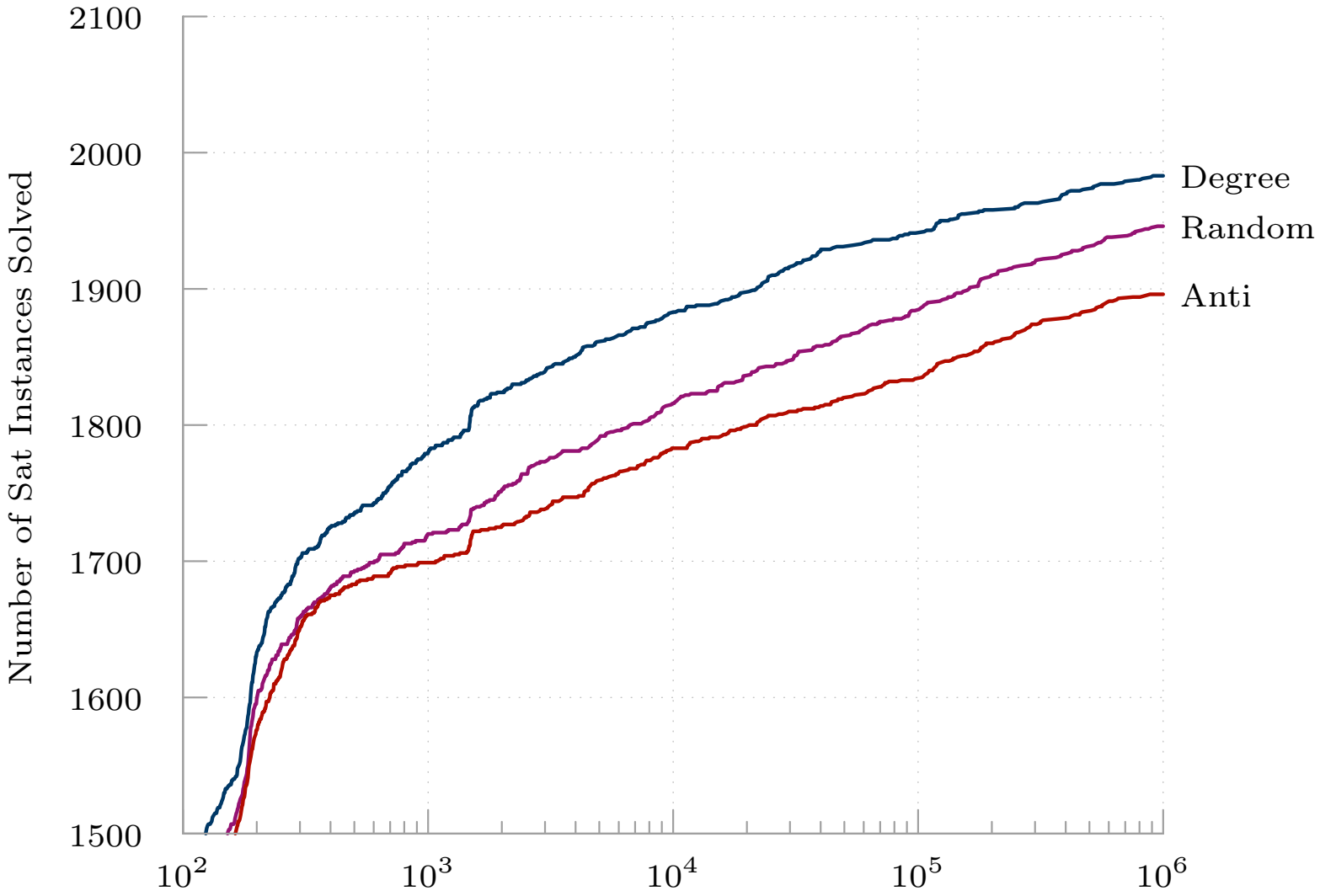




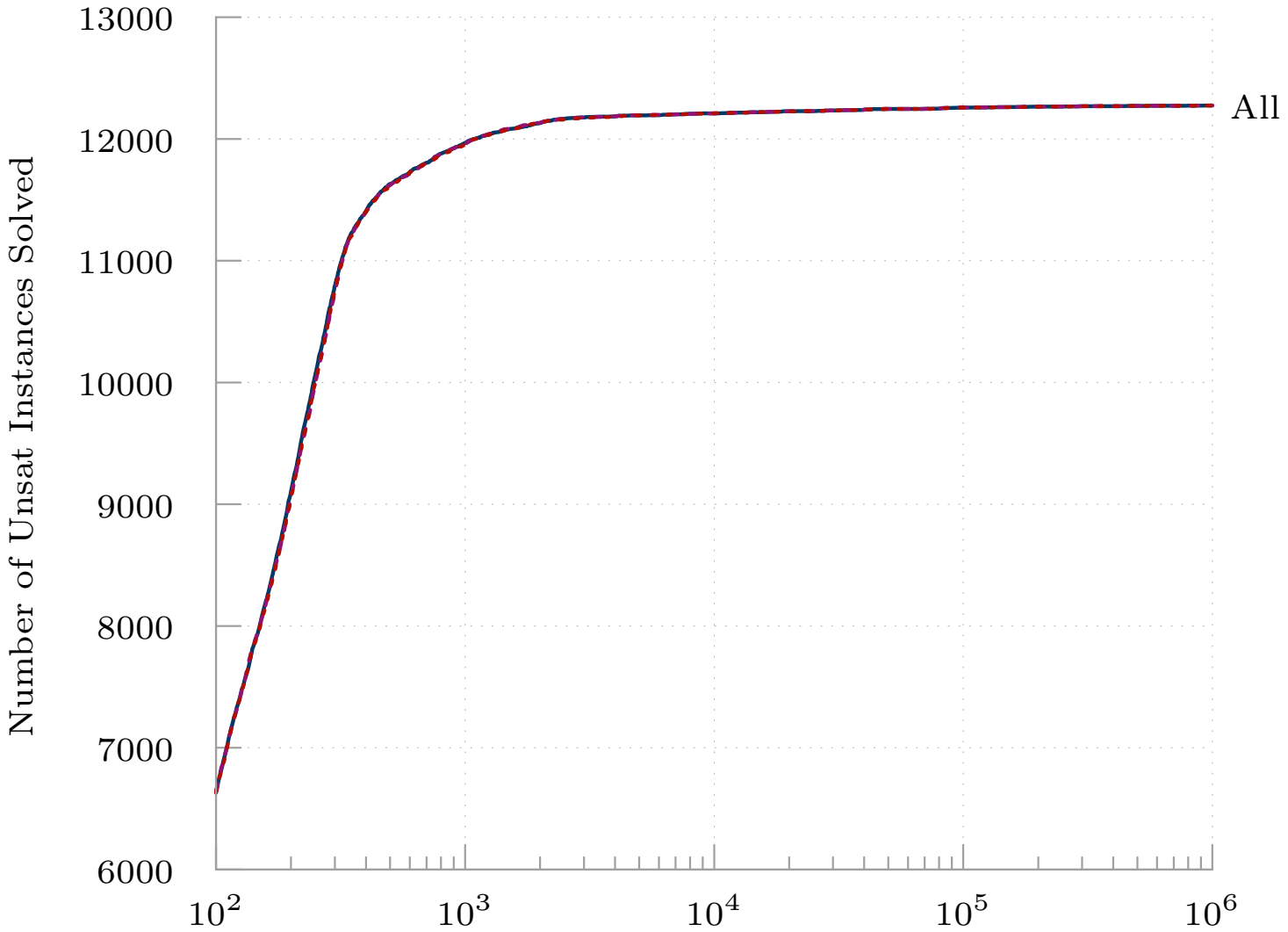
Subgraph Isomorphism

- A variable for each vertex in the pattern graph.
  - Smallest domain first.
  - Tiebreak on highest degree first.
- Domains are target vertices.
  - Highest degree to lowest.

Subgraph Isomorphism



Subgraph Isomorphism



Heuristics and Discrepancies

- If our value-ordering heuristics are perfect, and an instance is satisfiable, we walk straight to a solution by going left at every level.
- If an instance is unsatisfiable, perfect variable-ordering heuristics would give the smallest possible search tree.
- But heuristics aren't perfect...
- We call going against a value-ordering heuristic choice a "discrepancy".

Two Claims About Value-Ordering Heuristics

Limited Discrepancy Search

William D. Harvey and **Matthew L. Ginsberg**
CIRL
1269 University of Oregon
Eugene, Oregon 97403
U.S.A.
ginsberg@cs.uoregon.edu

1. The total number of discrepancies to find a solution is usually low (our value-ordering heuristics are *usually* right).
2. Value-ordering heuristics are most likely to wrong higher up in the tree (there is least information available when no or few choices have been made).

So What?

- If these claims are true, depth-first search is a bad idea: we're committing entirely to the first decision made, which is most likely to be wrong.

Limited Discrepancy Search

- First, search with no discrepancies.
- Then search allowing one discrepancy.
  - First try one discrepancy at the top.
  - Then try one discrepancy at the second level.
  - Then try one discrepancy at the third level.
  - ...
- Then search allowing two discrepancies.
  - At the top, and at the second level.
  - Then at the top, and at the third level.
  - ...
  - Then at the second level and the third level.
  - ...
- ...

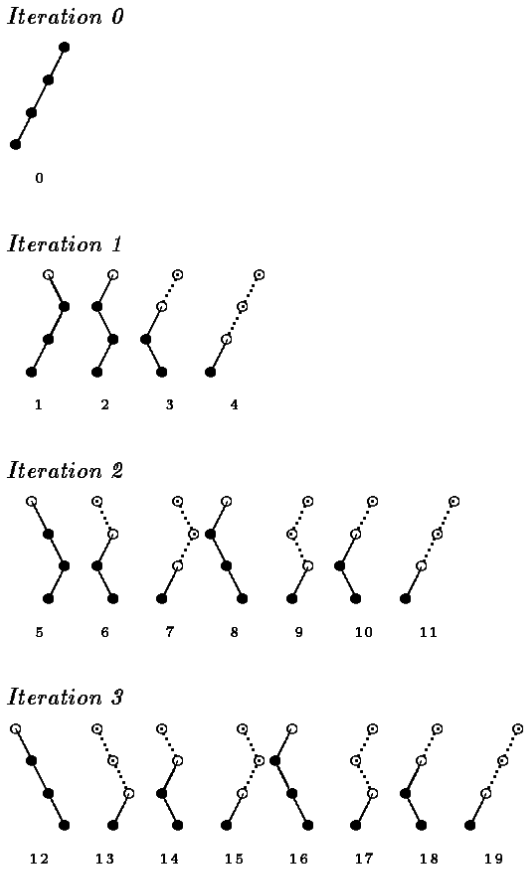


Figure 2: Execution trace of LDS.

Completeness

- Complete: yes means yes, no means no.
- Incomplete: yes means yes, no means maybe.
- LDS is *quasi-complete*: if the total number of discrepancies is allowed to go high enough, it is complete.
- Discrepancy searches do more total work if there is no solution, and make optimality proofs longer.

Non-Binary Trees?

- We can rewrite our search tree to be binary. Instead of branching on each value for a variable in a loop, pick a variable and a value, and branch twice:
  - Yes, the variable takes that value.
  - No, the variable does not take that value.
- But this means that giving the 10th value to a variable counts as 9 discrepancies. Is this good or bad?
- Alternatively, we can treat the left branch as no discrepancy, and all right branches as discrepancies.

Using LDS?

[gecode-users] Important: Licensing information regarding LDS

Christian Schulte cschulte at kth.se
Sat Oct 9 20:35:51 CEST 2010

- Previous message: [gecode-users] Gecode 3.4.2 released
- **Messages sorted by:** [date] [thread] [subject] [author]

```
Dear all,

We have been informed by one of the patent holders that LDS (limited
discrepancy search) is patented in the United States of America. While this
does not pose an issue per se for us as developers (the MIT license under
which Gecode is released makes that clear), it does to you as users.

After weighing the merits of offering LDS in Gecode with the effort for
obtaining a non-commercial license, we have decided to remove LDS from
Gecode. Gecode 3.4.2 removes LDS.

This decision just reflects our current understanding of how useful LDS is
compared to any effort regarding licensing.

If you feel strongly about having LDS for Gecode available, we might make it
available as an additional contribution with an explicit statement that it
is patented in the United States of America. The patent holder has informed
me that he is willing to give a non-commercial license to anybody who seeks
one.

Please also take this information into account when using versions of Gecode
before 3.4.2: you need to have a license to use LDS in the United States of
America.

Christian
```

Depth-Bounded Discrepancy Search

Depth-bounded Discrepancy Search

Toby Walsh*
APES Group, Department of Computer Science
University of Strathclyde, Glasgow G1 1XL. Scotland
tw@cs.strath.ac.uk

- If the second claim *is* important, why not emphasise it more?
- Depth-bounded discrepancy search considers k discrepancies, but only at depth up to $k-1$.

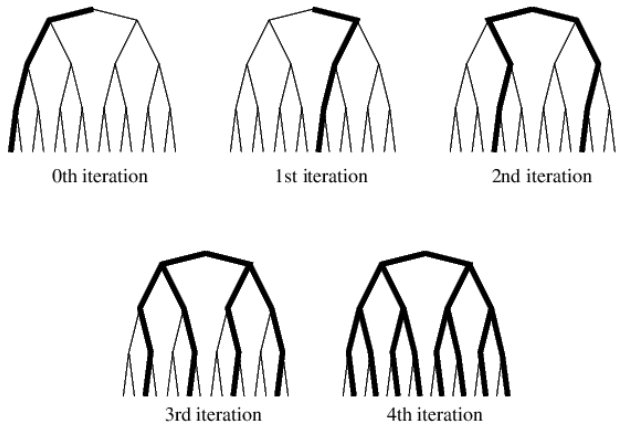


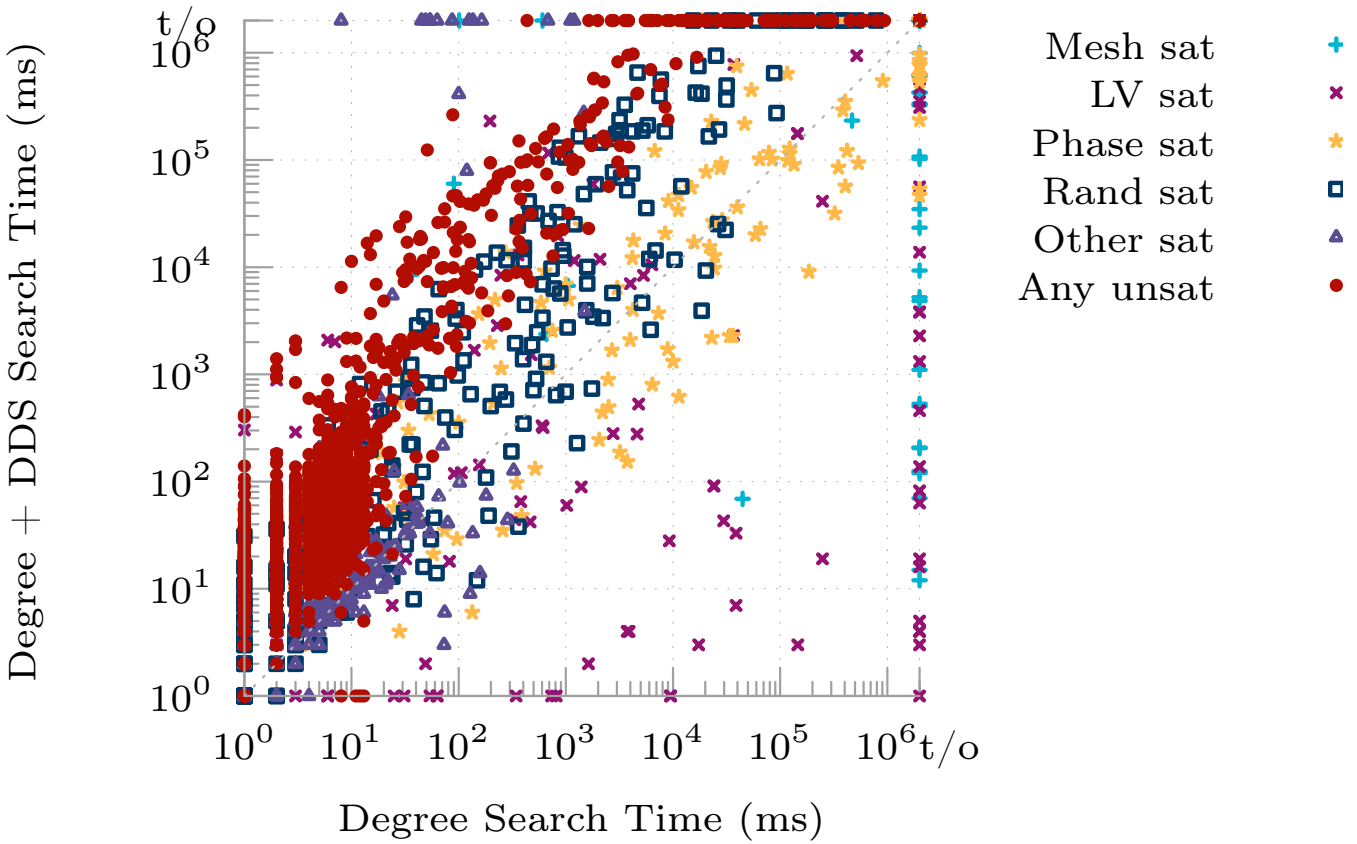
Figure 2: DDS on a binary tree of depth 4.

Depth-Bounded Discrepancy Search

```
function DDS
  k = 0
  repeat
      ⟨goal, depth⟩ = PROBE(root,k)
      k = k + 1
  until goal or k > depth
  return goal
```

```
function PROBE(node,k)
  if leaf(node) then return ⟨goal-p(node), 0⟩
  if k = 0 then
      ⟨goal, depth⟩ = PROBE(left(node),0)
      return ⟨goal, 1 + depth⟩
  if k = 1 then
      ⟨goal, depth⟩ = PROBE(right(node),0)
      return ⟨goal, 1 + depth⟩
  if k > 1 then
      ⟨goal_1, depth_1⟩ = PROBE(left(node),k − 1)
      if goal_1 then return ⟨goal_1, 1 + depth_1⟩ else
          ⟨goal_2, depth_2⟩ = PROBE(right(node),k − 1)
          return ⟨goal_2, 1 + max(depth_1, depth_2)⟩
```

DDS for Subgraph Isomorphism



Restarts?

Sequential and Parallel Solution-Biased Search for Subgraph Algorithms*

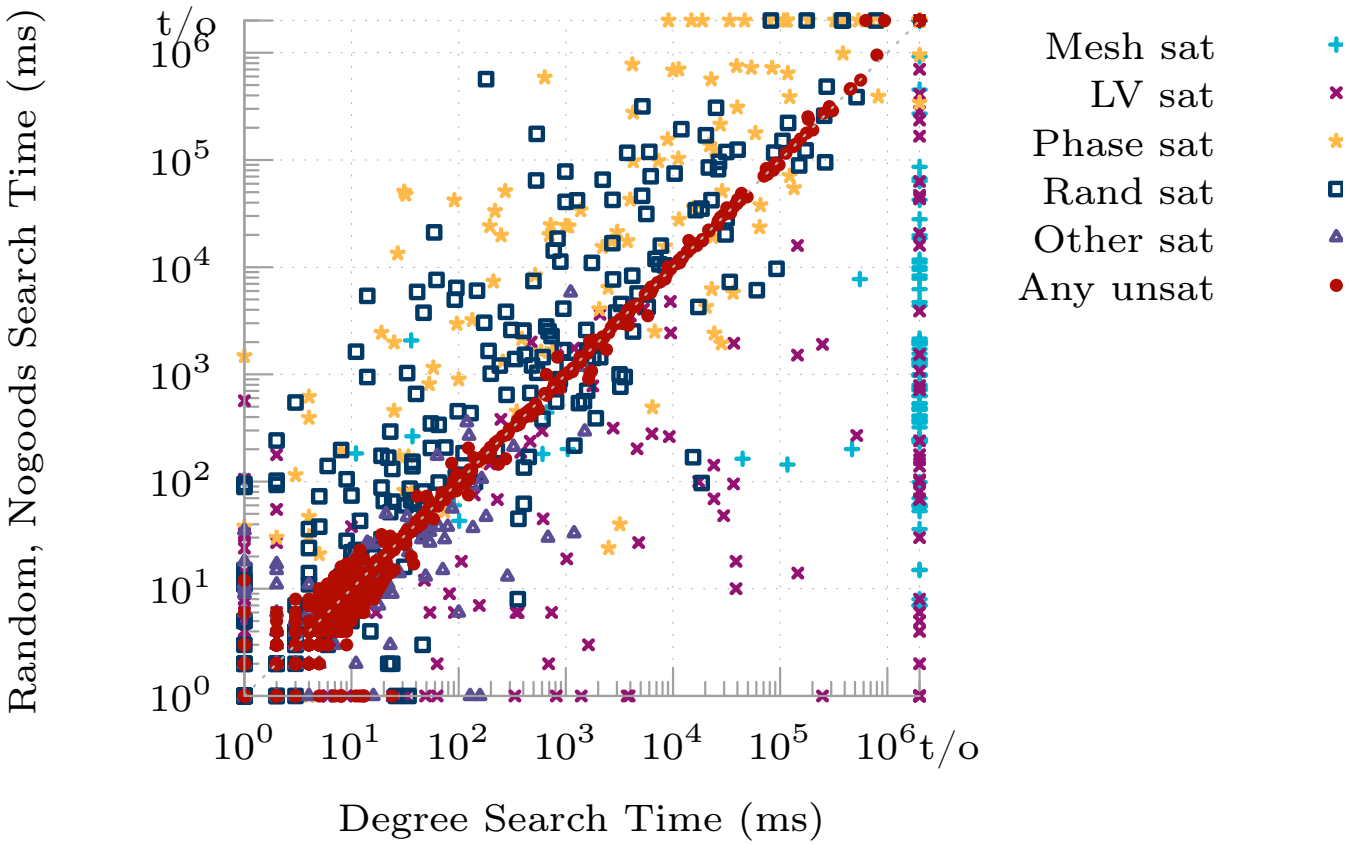
Blair Archibald[1 [0000−0003−3699−6658]], Fraser Dunlop[2 [0000−0002−4485−4871]], Ruth Hoffmann[2 [0000−0002−1011−5894]], Ciaran McCreesh[1 [0000−0002−6106−4871]], Patrick Prosser[1 [0000−0003−4460−6912]], and James Trimble[1 [0000−0001−7282−8745]]

[1] University of Glasgow, Glasgow, Scotland
[2] University of St Andrews, St Andrews, Scotland

Restarts?

- Run for a bit, then restart and try something else.
- Use nogood recording to avoid revisiting parts of the search space.
- Need to change something when we restart: what about just using a random value-ordering heuristic?

Restarts?

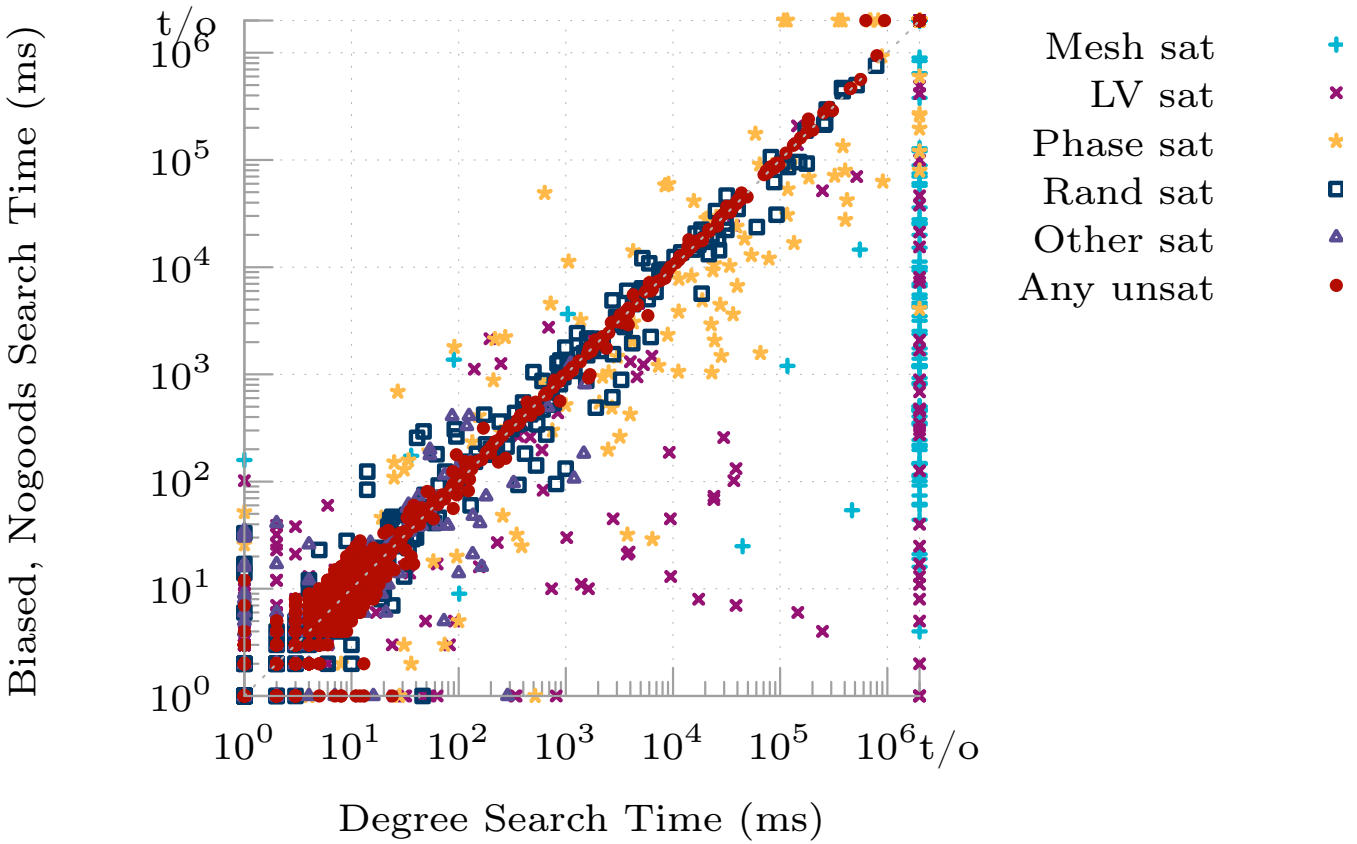

Random, Nogoods Search Time (ms)
t/o
10^6
10^5
10^4
10^3
10^2
10^1
10^0
10^0 10^1 10^2 10^3 10^4 10^5 10^6t/o
Degree Search Time (ms)
Mesh sat
LV sat
Phase sat
Rand sat
Other sat
Any unsat

Restarts?

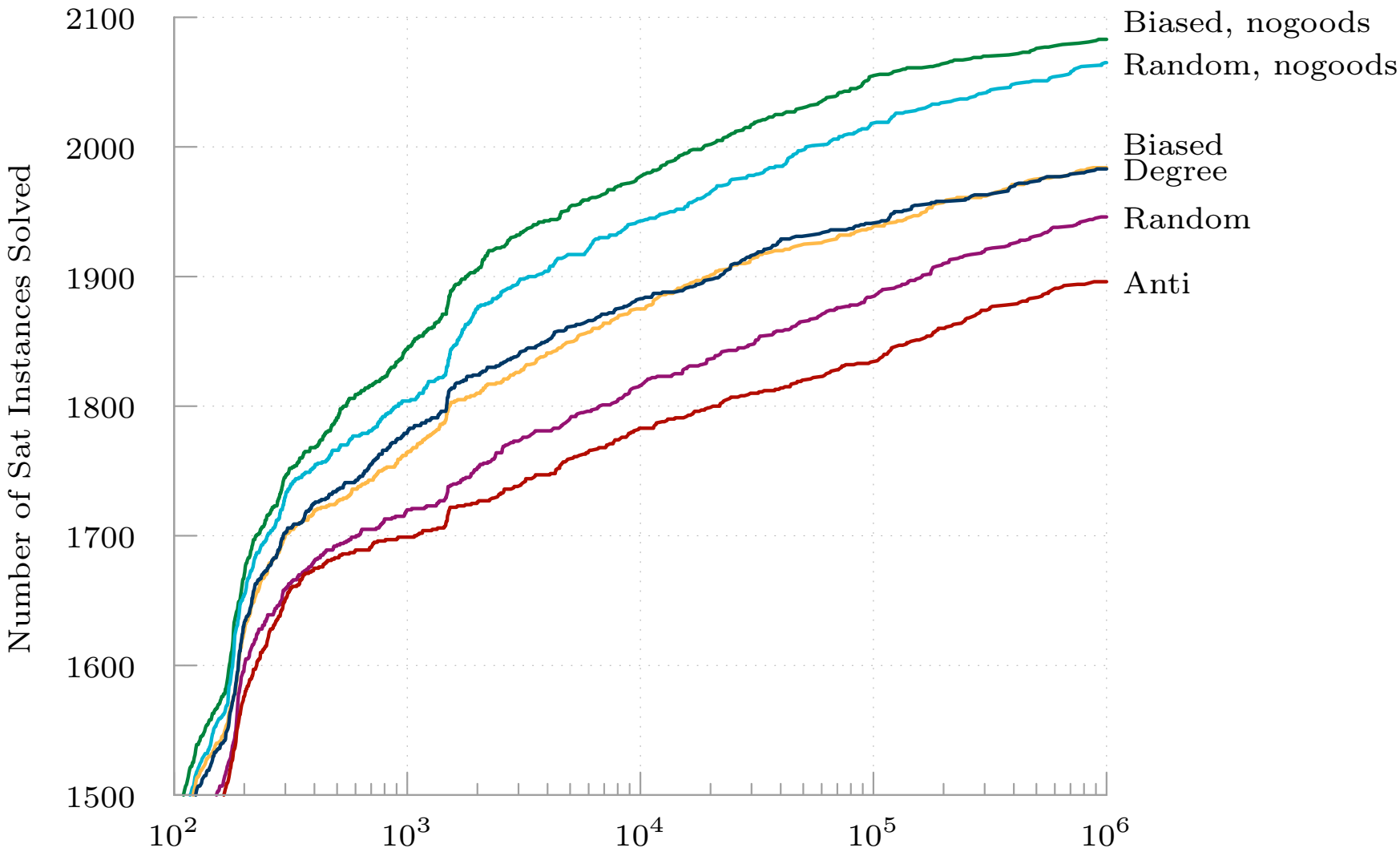
- Select a vertex v' from the chosen domain D_v with probability

$$p(v') = \frac{2^{\deg(v')}}{\sum_{w \in D_v} 2^{\deg(w)}}.$$

Restarts?


Biased, Nogoods Search Time (ms)
t/o
10^6
10^5
10^4
10^3
10^2
10^1
10^0
10^0 10^1 10^2 10^3 10^4 10^5 10^6t/o
Degree Search Time (ms)
Mesh sat
LV sat
Phase sat
Rand sat
Other sat
Any unsat

Restarts?


Number of Sat Instances Solved
2100
2000
1900
1800
1700
1600
1500
10^2
10^3
10^4
10^5
10^6
Biased, nogoods
Random, nogoods
Biased
Degree
Random
Anti

This is Not The Exam Question

What are the two assumptions regarding value ordering heuristics which underlie limited discrepancy search?

Why are discrepancy searches a bad choice if instances are expected to be unsatisfiable?

When using MAC, what effect do value ordering heuristics have on unsatisfiable instances?