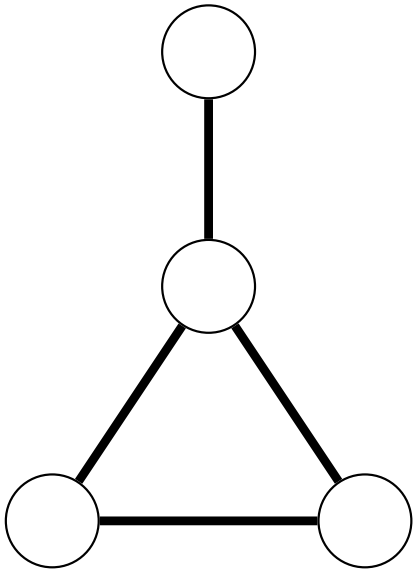
Subgraph Isomorphism in Practice

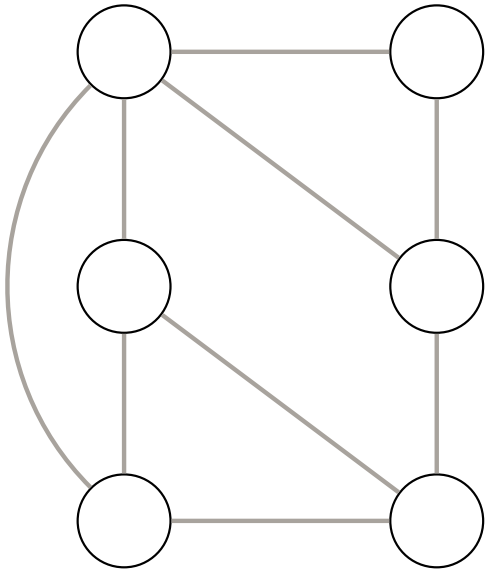
Ciaran McCreesh

in conspiracy with Blair Archibald, Fraser Dunlop, Stephan Gocht, Ruth Hoffmann, Jakob Nordström, Patrick Prosser, Christine Solnon and James Trimble

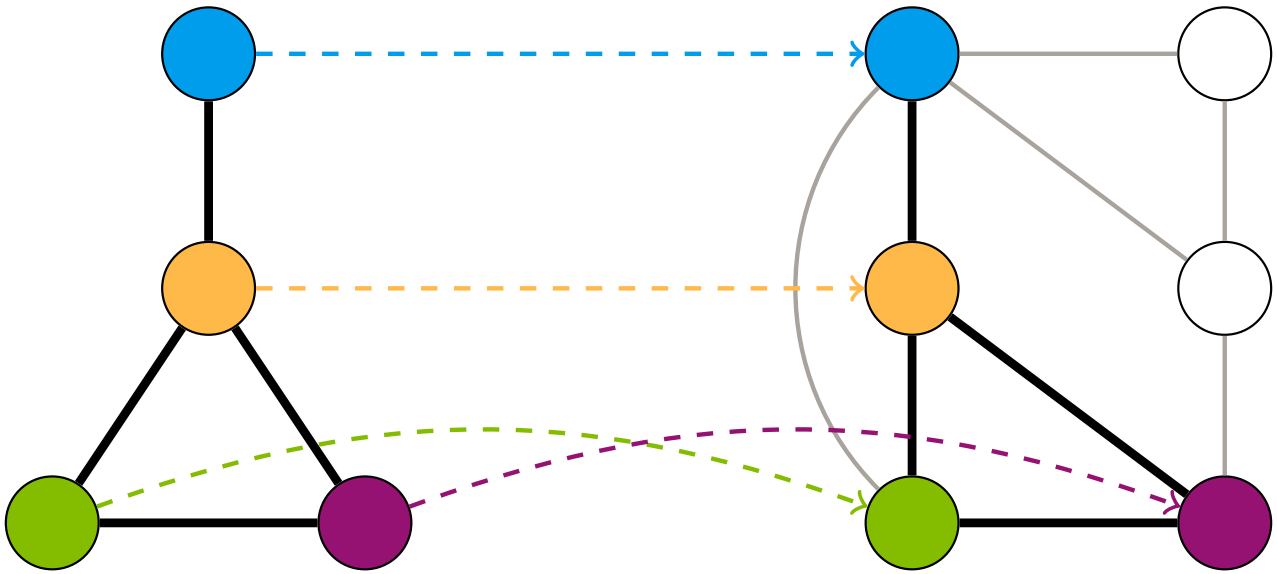
University of Glasgow

Subgraph Isomorphism

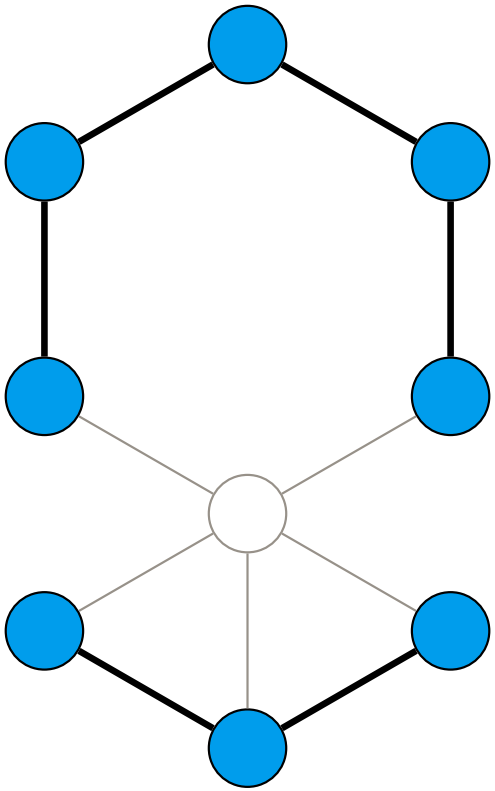


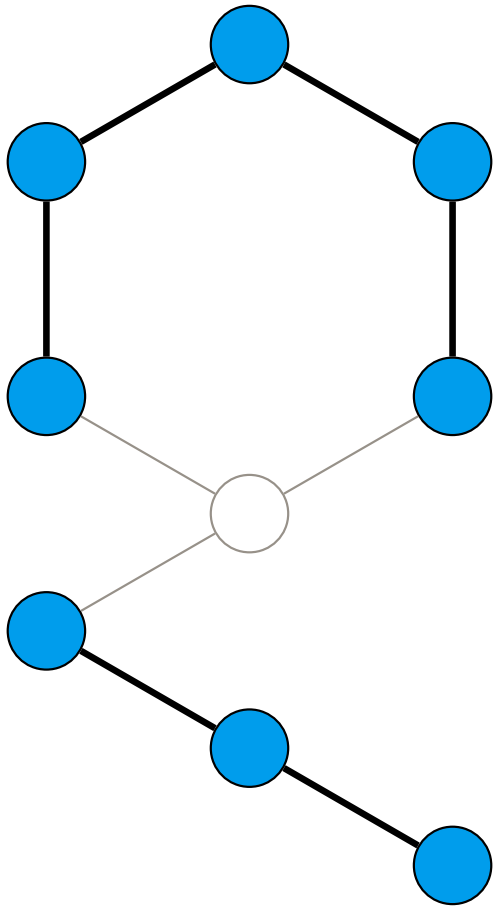


Subgraph Isomorphism

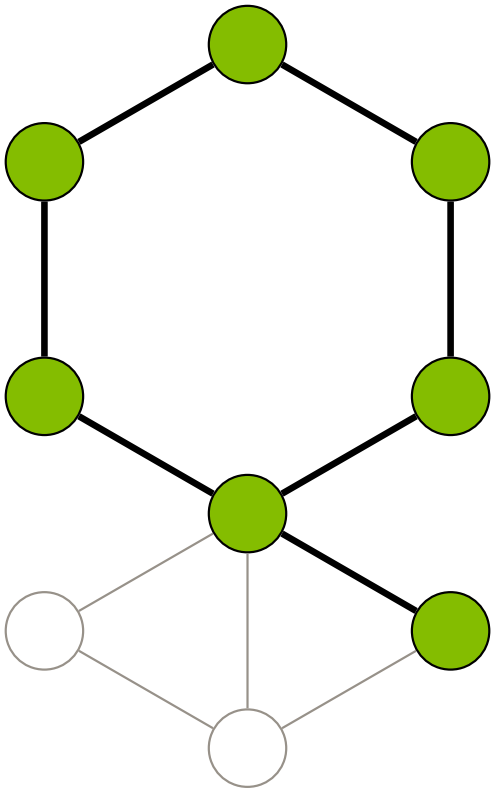


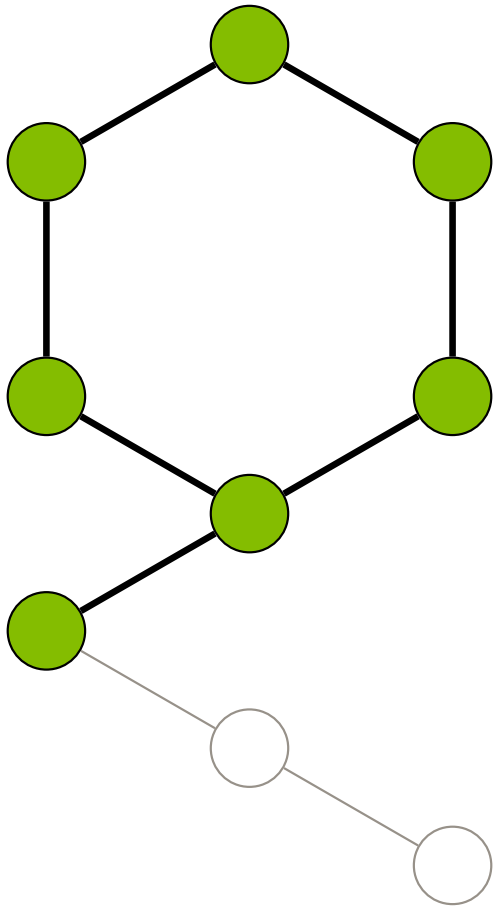
Maximum Common Induced Subgraph



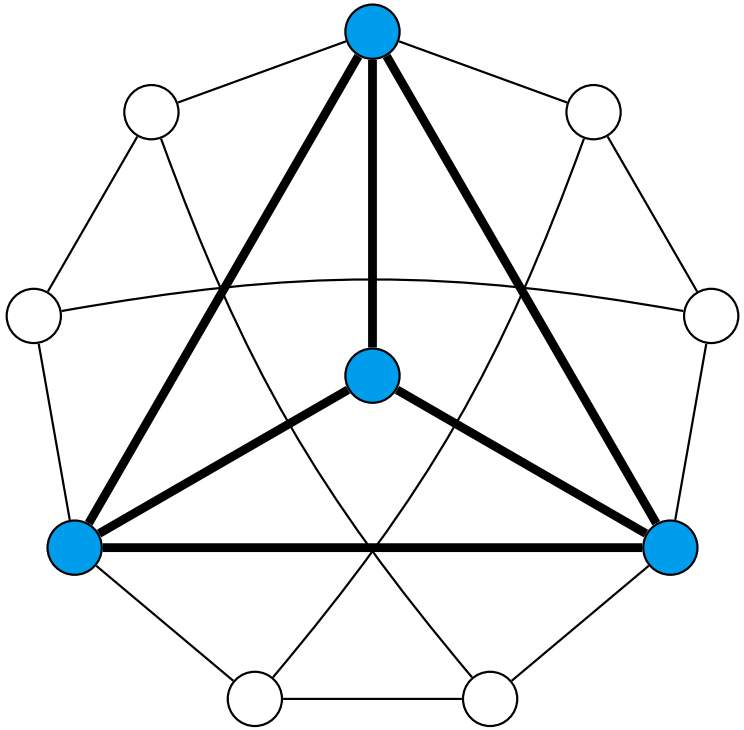


Maximum Common Induced Connected Subgraph





Maximum Clique



Who Cares?

- Chemistry, biochemistry, and drug design (graphs are molecule fragments or proteins).
- Computer vision.
- Compilers (instruction generation, code rewriting).
- Plagiarism and malware detection.
- Livestock epidemiology (contact and trade graphs).
- Designing mechanical lock systems.

In Theory…

- Subgraph finding is hard.
- Subgraph counting is hard.
- Approximate subgraph finding is hard.

In Practice...

- We have good *solvers* for subgraph problems.
- Some applications involve solving thousands of subgraph isomorphism queries per second.
- We can solve clique on larger graphs than we can solve all-pairs shortest path.[1]

[1] Terms and conditions apply.

In Practice…

- We have good *solvers* for subgraph problems.
- Some applications involve solving thousands of subgraph isomorphism queries per second.
- We can solve clique on larger graphs than we can solve all-pairs shortest path.[1]
- Maximum common subgraph is still a nightmare…

[1] Terms and conditions apply.

Subgraph Isomorphism, as a Constraint Program

- A variable for each pattern vertex. The domains are all of the target vertices.
- At least two sets of constraints:
  - Adjacent pairs of vertices must be mapped to adjacent pairs of vertices.
  - All different.
- Then we get clever:
  - Extra constraints about degrees, paths, ...
  - Better data structures and propagation queues.
  - Very good variable- and value-ordering heuristics.

The Glasgow Subgraph Solver

`https://github.com/ciaranm/glasgow-subgraph-solver`

- A CP style solver specifically for subgraph algorithms.
- Subgraph isomorphism, and all its variants (induced / non-induced, homomorphism, locally injective, labels, side constraints, directed, …).
- Also special algorithms for clique.
- Guaranteed no bugs!

The Glasgow Subgraph Solver

`https://github.com/ciaranm/glasgow-subgraph-solver`

- A CP style solver specifically for subgraph algorithms.
- Subgraph isomorphism, and all its variants (induced / non-induced, homomorphism, locally injective, labels, side constraints, directed, …).
- Also special algorithms for clique.
- Guaranteed no bugs!
  - Or at least, any buggy output will always be detected, if you enable proof logging.

Benchmark Instances

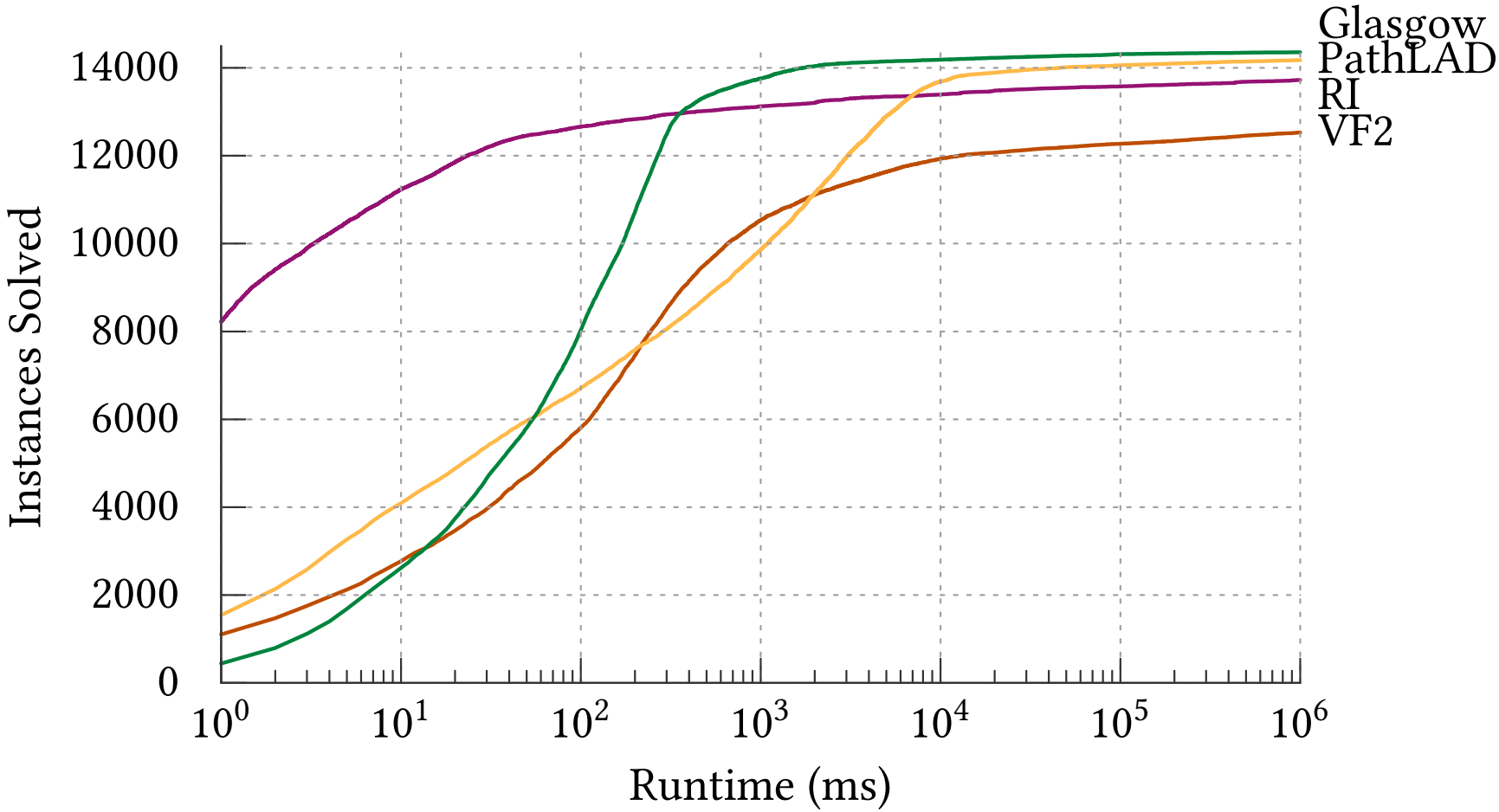
- 14,621 instances from Christine Solnon's collection:
  - Randomly generated with different models.
  - Real-world graphs.
  - Computer vision problems.
  - Biochemistry problems.
  - Phase transition instances.
- At least...
  - $\geq$ 2,110 satisfiable.
  - $\geq$ 12,322 unsatisfiable.
- A lot of them are very easy for good algorithms.

Hardware

- HPC, optimised for throughput not reproducibility.[2]
- Dual Intel Xeon E5-2695 v4 CPUs, 2×18 cores
- 256GBytes RAM
- GCC 7.2.0
- C++ native threads, SGI MPT MPI

[2]This work used the Cirrus UK National Tier-2 HPC Service at EPCC (http://www.cirrus.ac.uk) funded by the University of Edinburgh and EPSRC (EP/P020267/1)

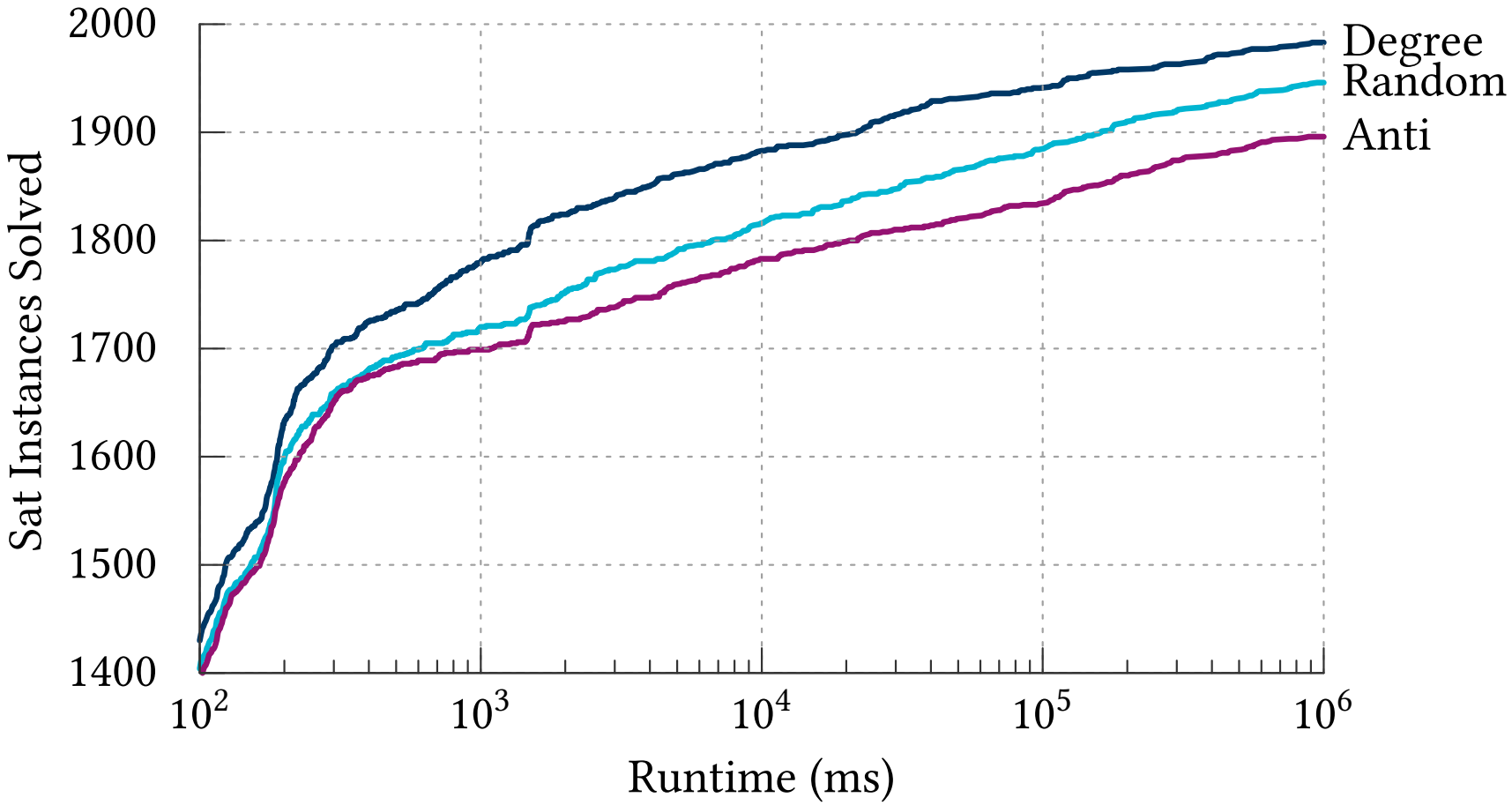
Is It Any Good?



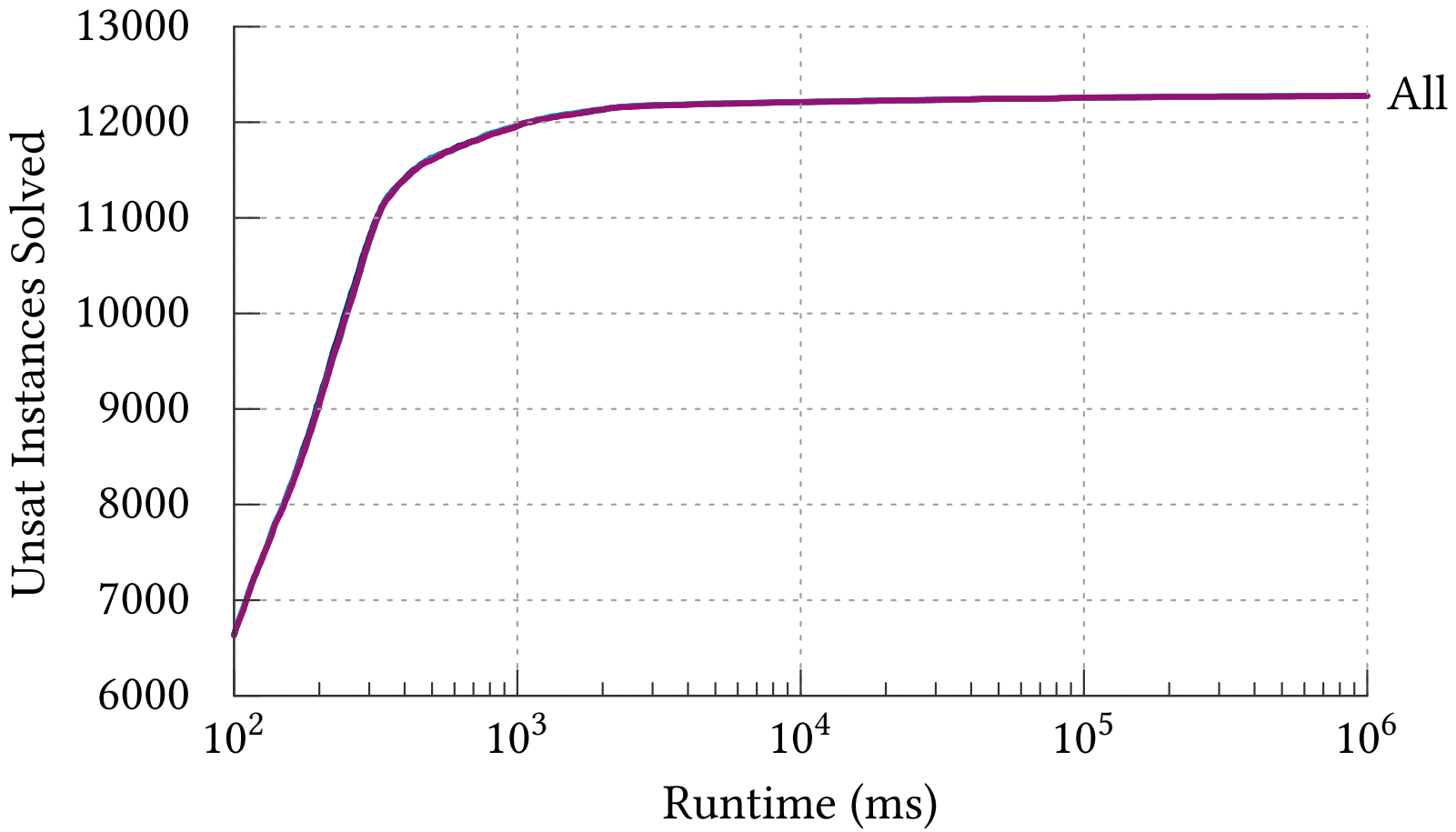
Search Order

- Variable ordering (i.e. pattern vertices): smallest domain first, tie-breaking on highest degree.
- Value ordering (i.e. target vertices): highest degree to lowest.

Sanity Check



Sanity Check



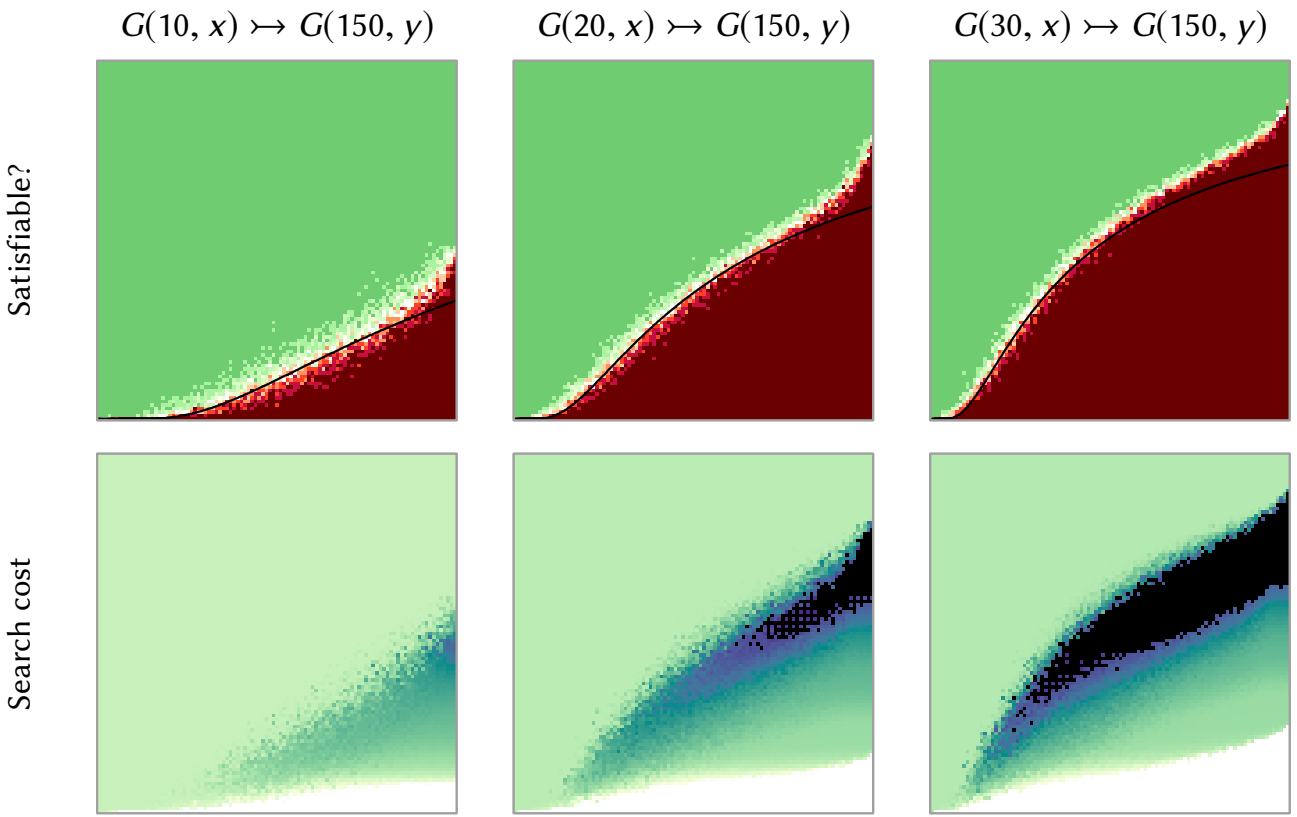
Hand-Wavy Theoretical Justification

- Maximise the expected number of solutions during search?
- If $P = G(p, q)$ and $T = G(t, u)$,

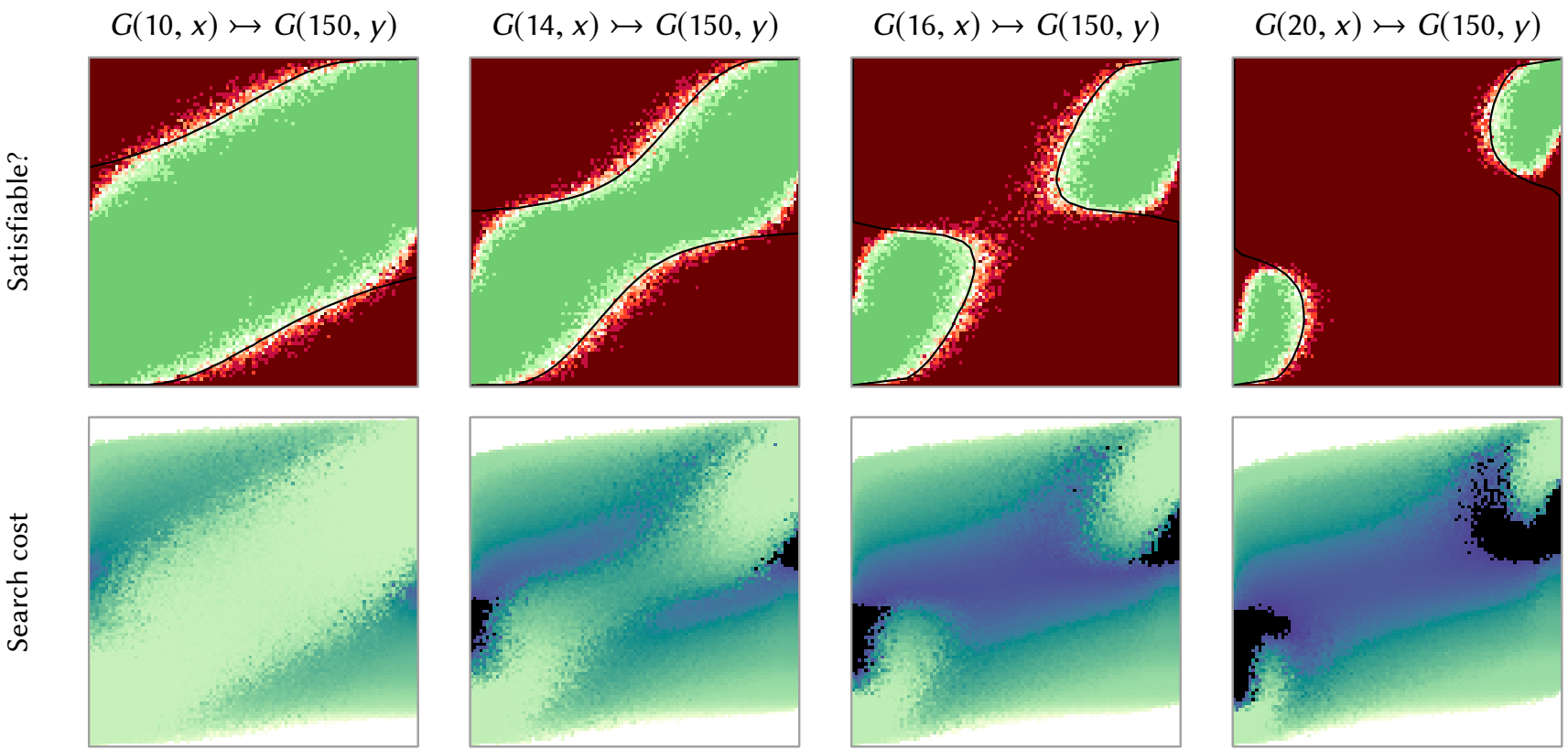
$$\langle Sol \rangle = \underbrace{t \cdot (t-1) \cdot \ldots \cdot (t-p+1)}_{\text{injective mapping}} \cdot \underbrace{u^{q \cdot \binom{p}{2}}}_{\text{adjacency}}$$

- Smallest domain first keeps remaining domain sizes large.
- High pattern degree makes the remaining pattern subgraph sparser, reducing q.
- High target degree leaves as many vertices as possible available for future use, making u larger.

Subgraph Isomorphism in Random Graphs



Induced Subgraph Isomorphism is More Complicated...



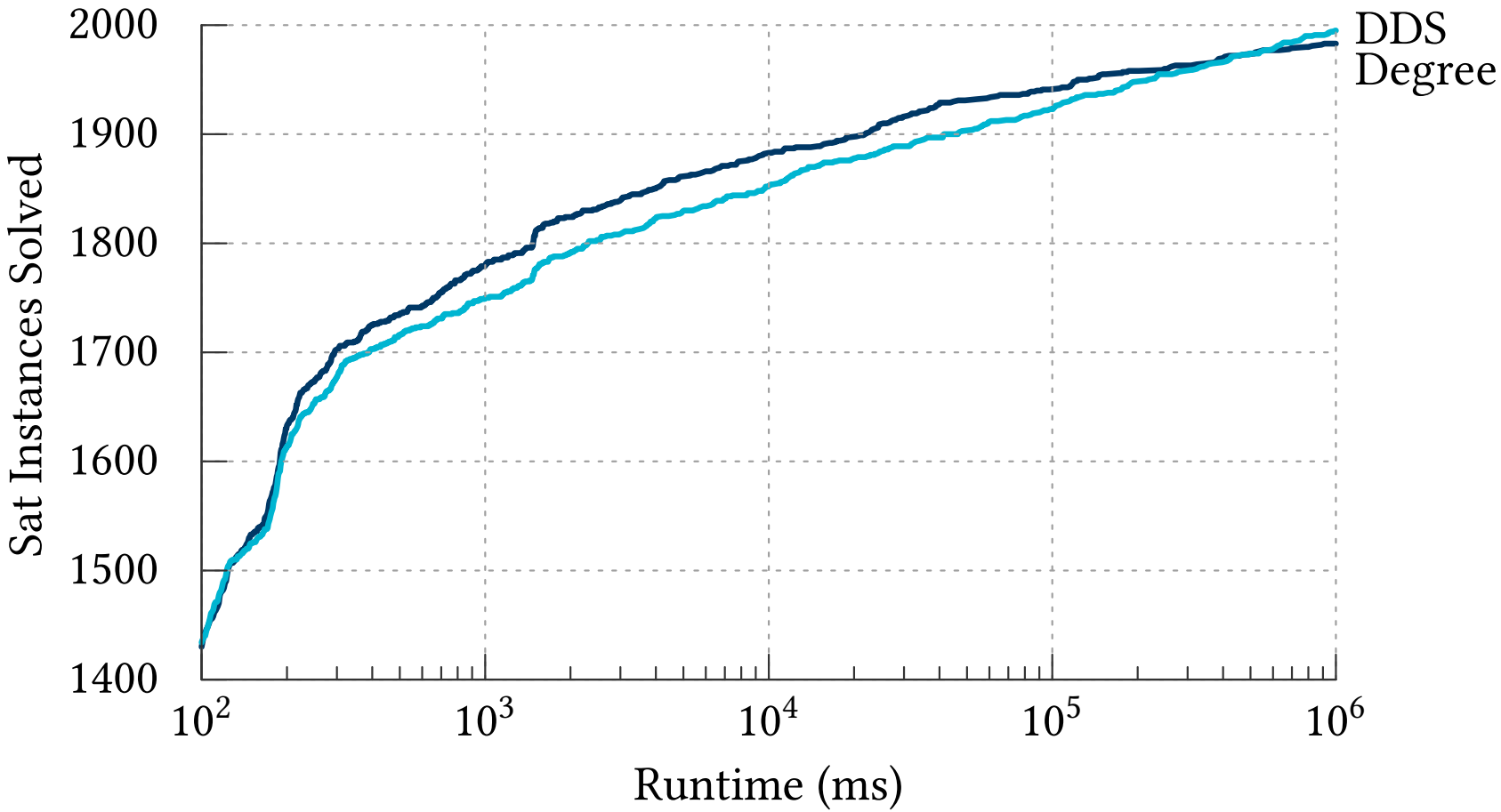
Back to Value-Ordering Heuristics

- Largest target degree first.

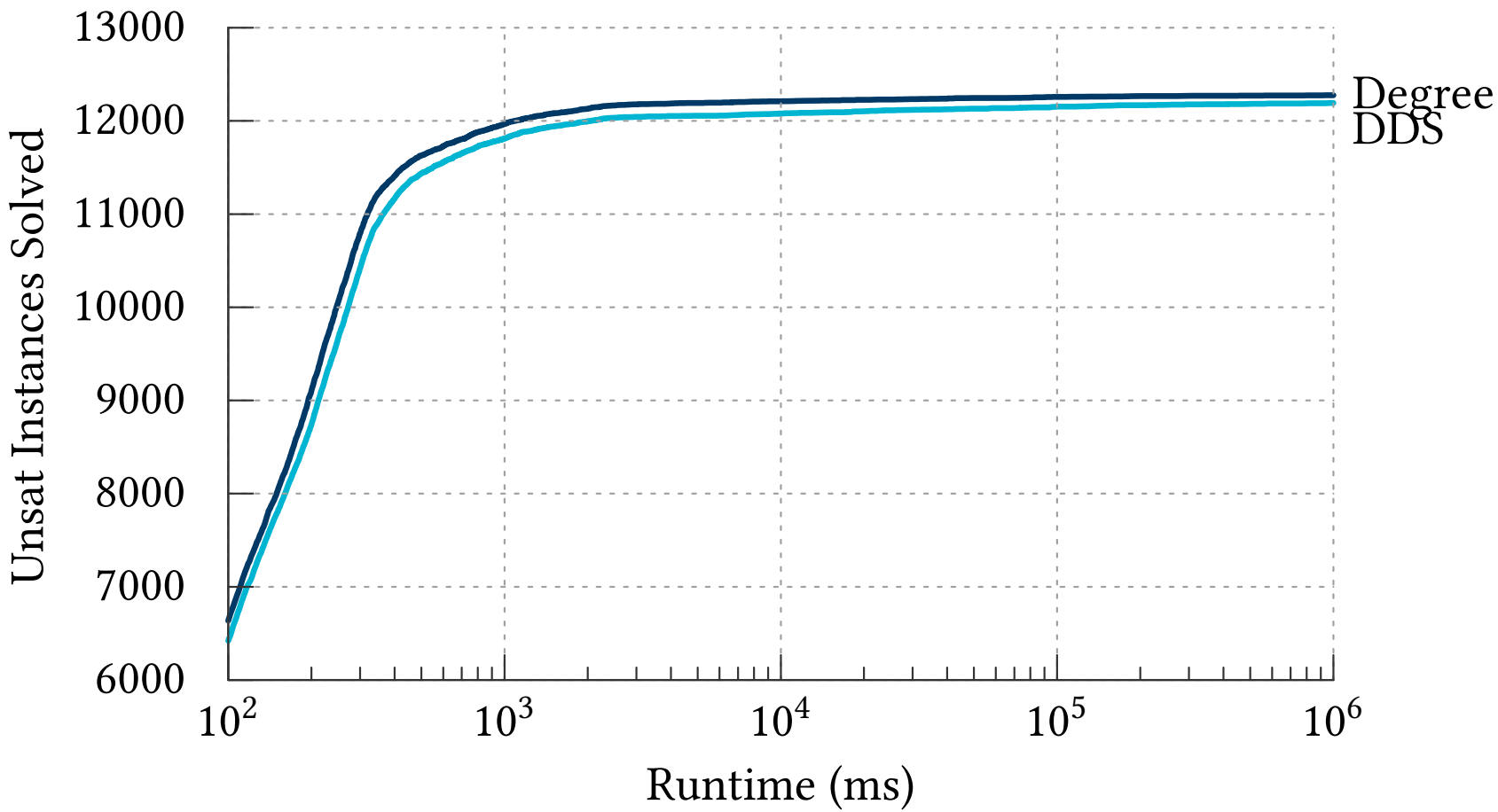
However…

- What if several vertices have the same degree?
- Is a vertex of degree 10 really that much better than a vertex of degree 9?

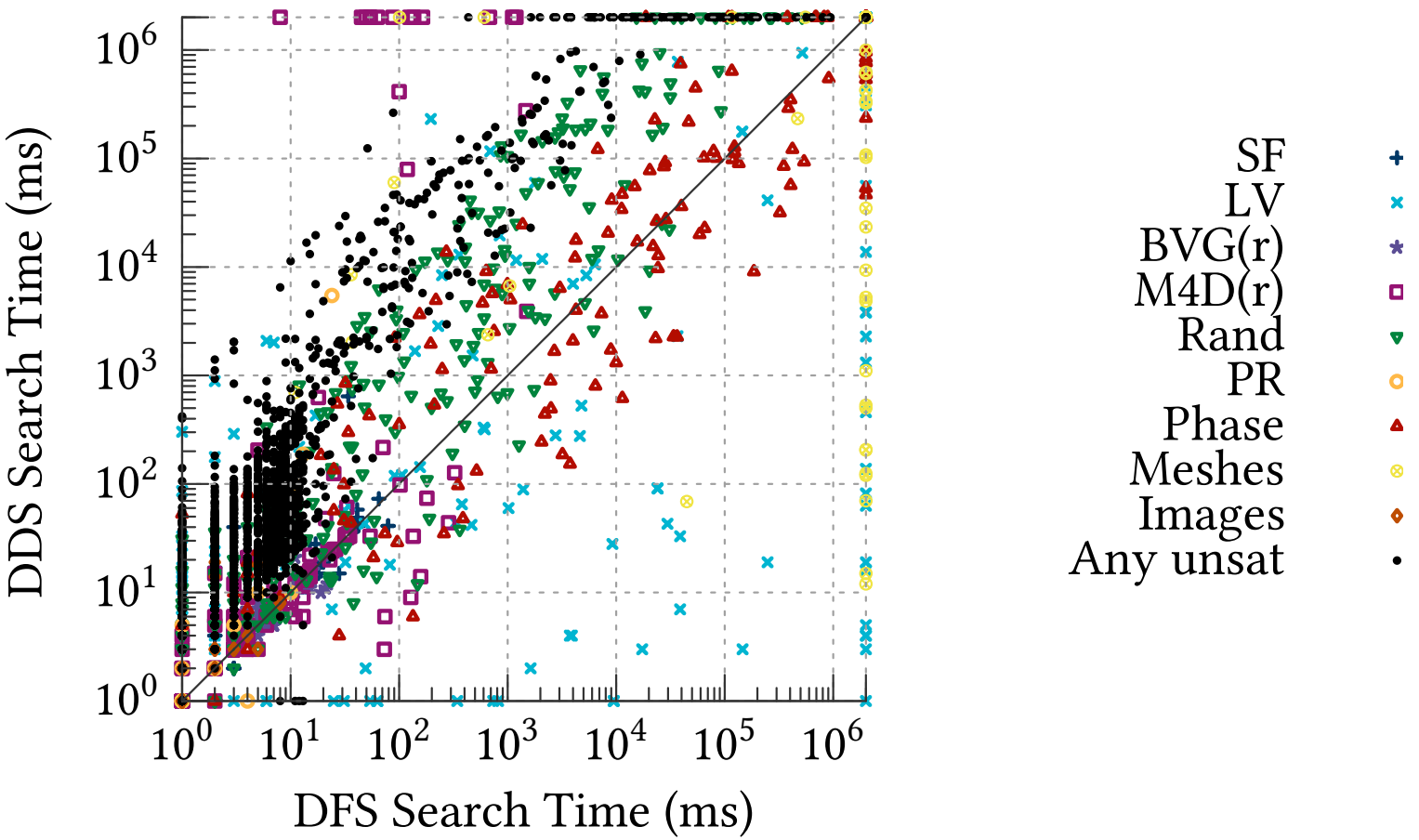
Discrepancy Search?



Discrepancy Search?



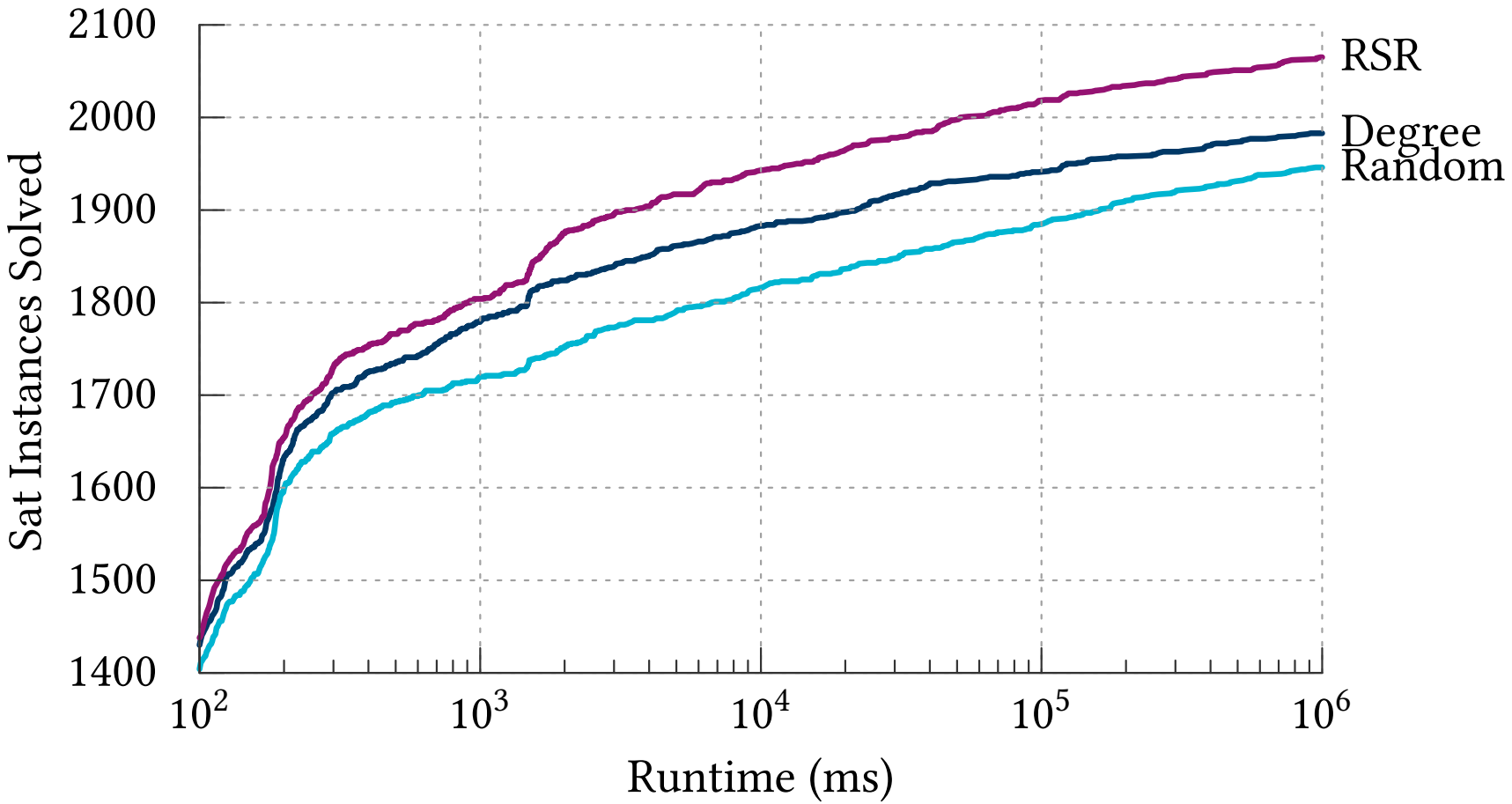
Discrepancy Search?



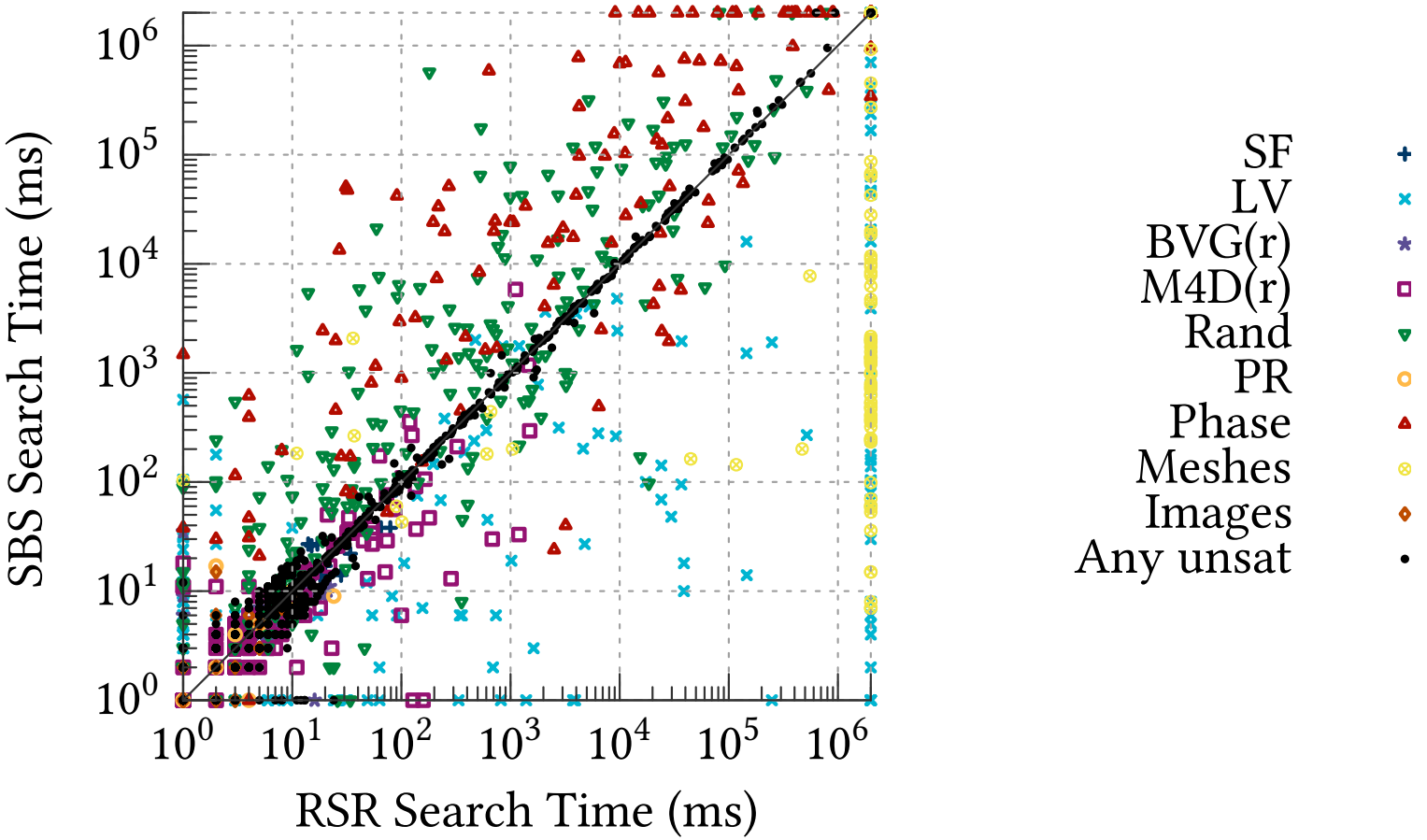
Random Search with Restarts and Nogood Recording

- Back to the random value-ordering heuristic.
- Aggressive restarts: every 100ms.
- Nogood recording and 2WL to avoid repeating work.

Random Search with Restarts and Nogood Recording



Random Search with Restarts and Nogood Recording



Value-Ordering Heuristics as Distributions

- Traditional view: value-ordering defines a search order.
- New view: value-ordering defines what proportion of the search effort should be spent on different subproblems.
- According to people who know more statistics than me, if solutions are uniformly distributed, then random search with restarts should be better than DFS.

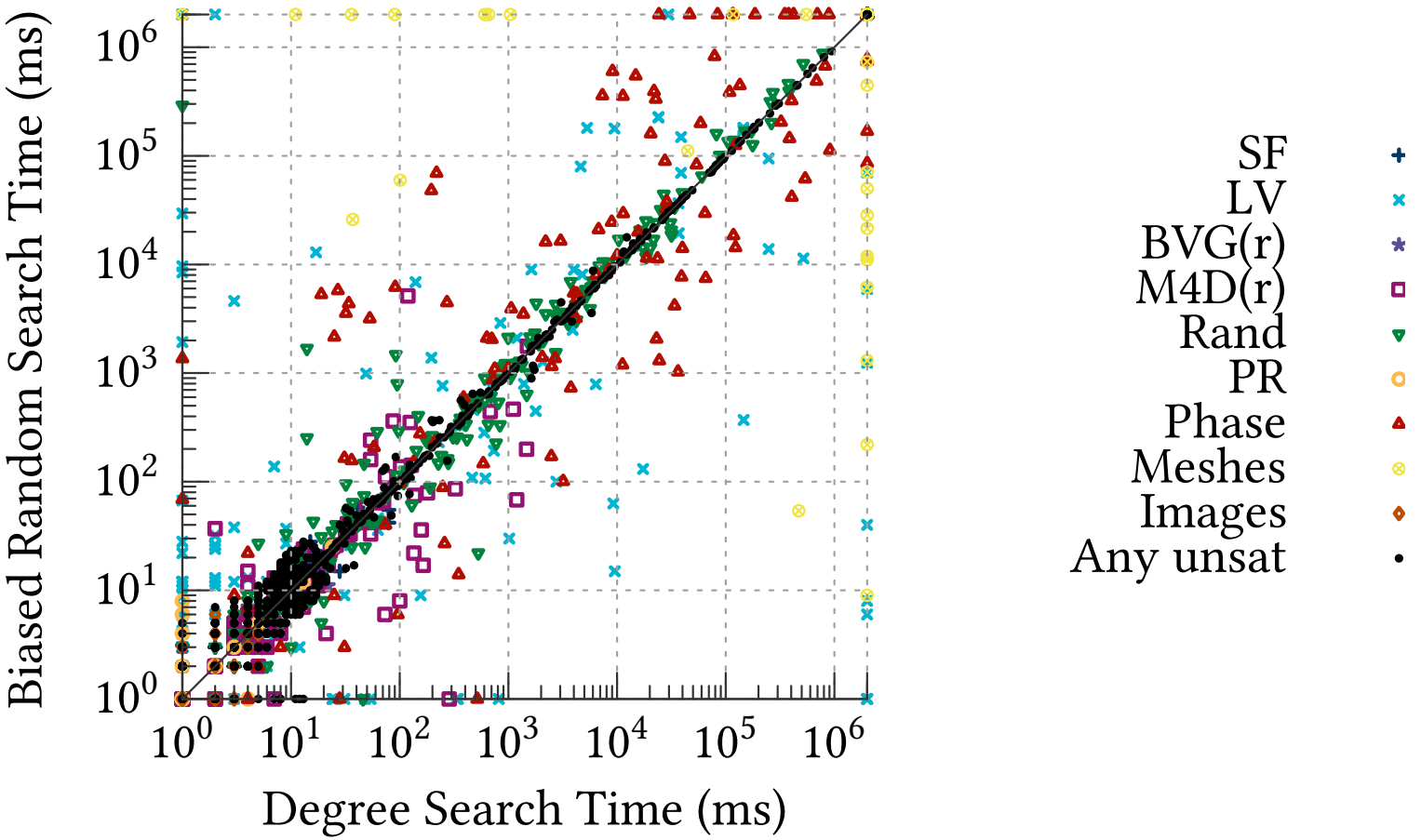
A Slightly Random Value-Ordering Heuristic

- For a fixed domain D_v, pick a vertex v' from a domain D_v with probability

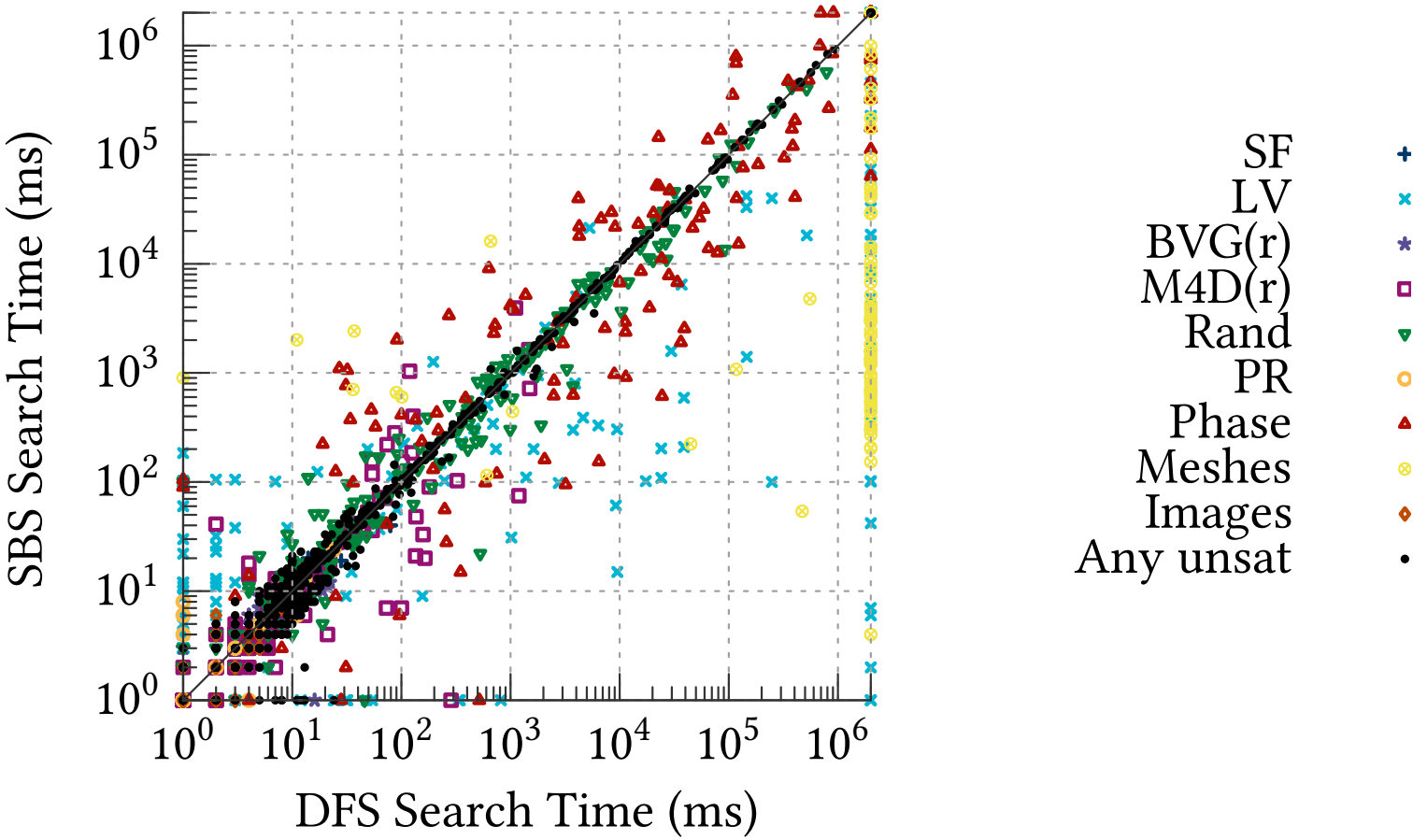
$$p(v') = \frac{2^{\deg(v')}}{\sum_{w \in D_v} 2^{\deg(w)}}$$

- Equally likely to pick between two vertices of degree d.
- Twice as likely to select a vertex of degree d than a vertex of degree $d - 1$.
- Justification: solution density and expected distribution of solutions.

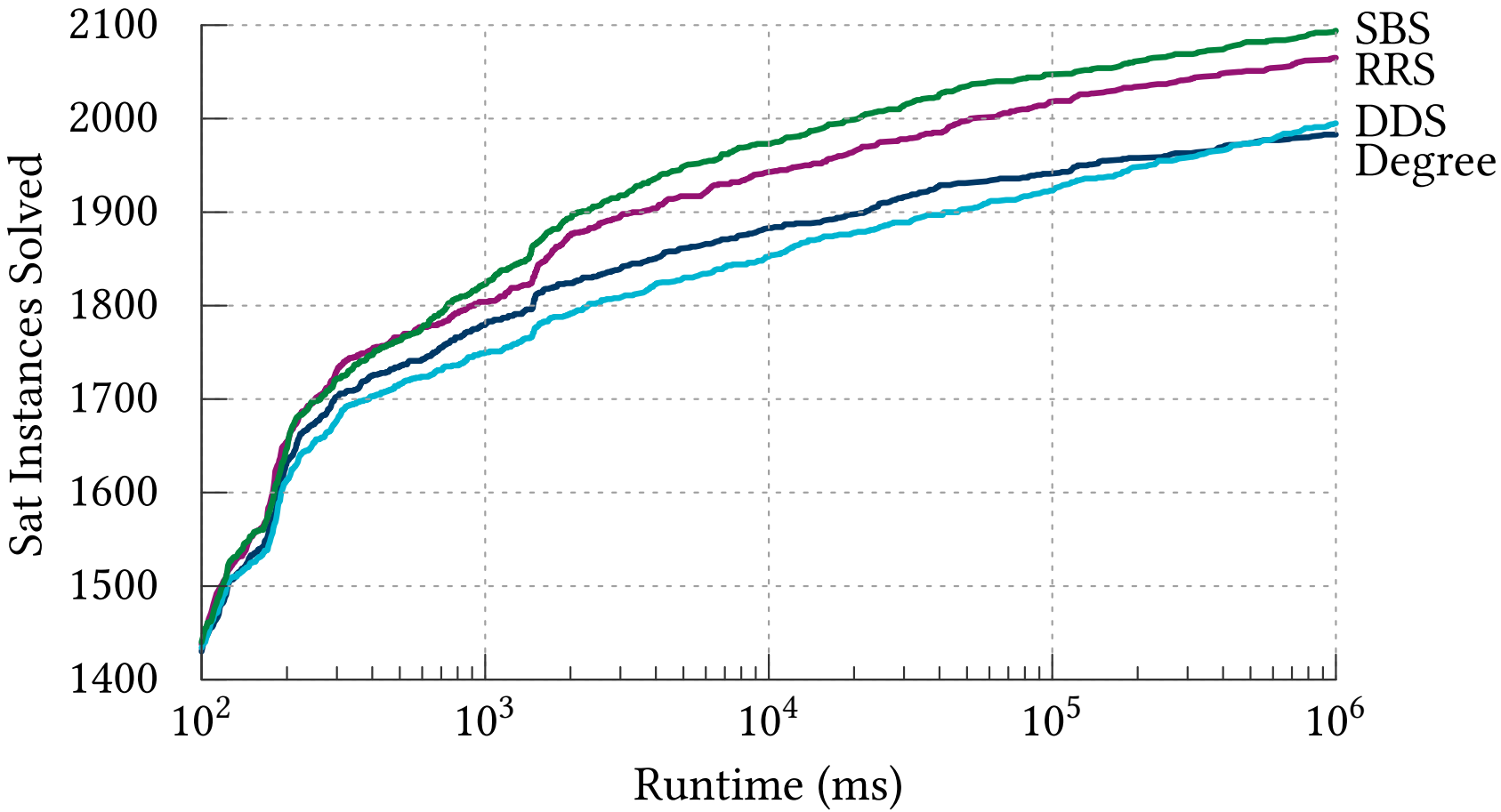
A Slightly Random Value-Ordering Heuristic



Is It Better?


SBS Search Time (ms)
DFS Search Time (ms)
10^0
10^1
10^2
10^3
10^4
10^5
10^6
SF
LV
BVG(r)
M4D(r)
Rand
PR
Phase
Meshes
Images
Any unsat

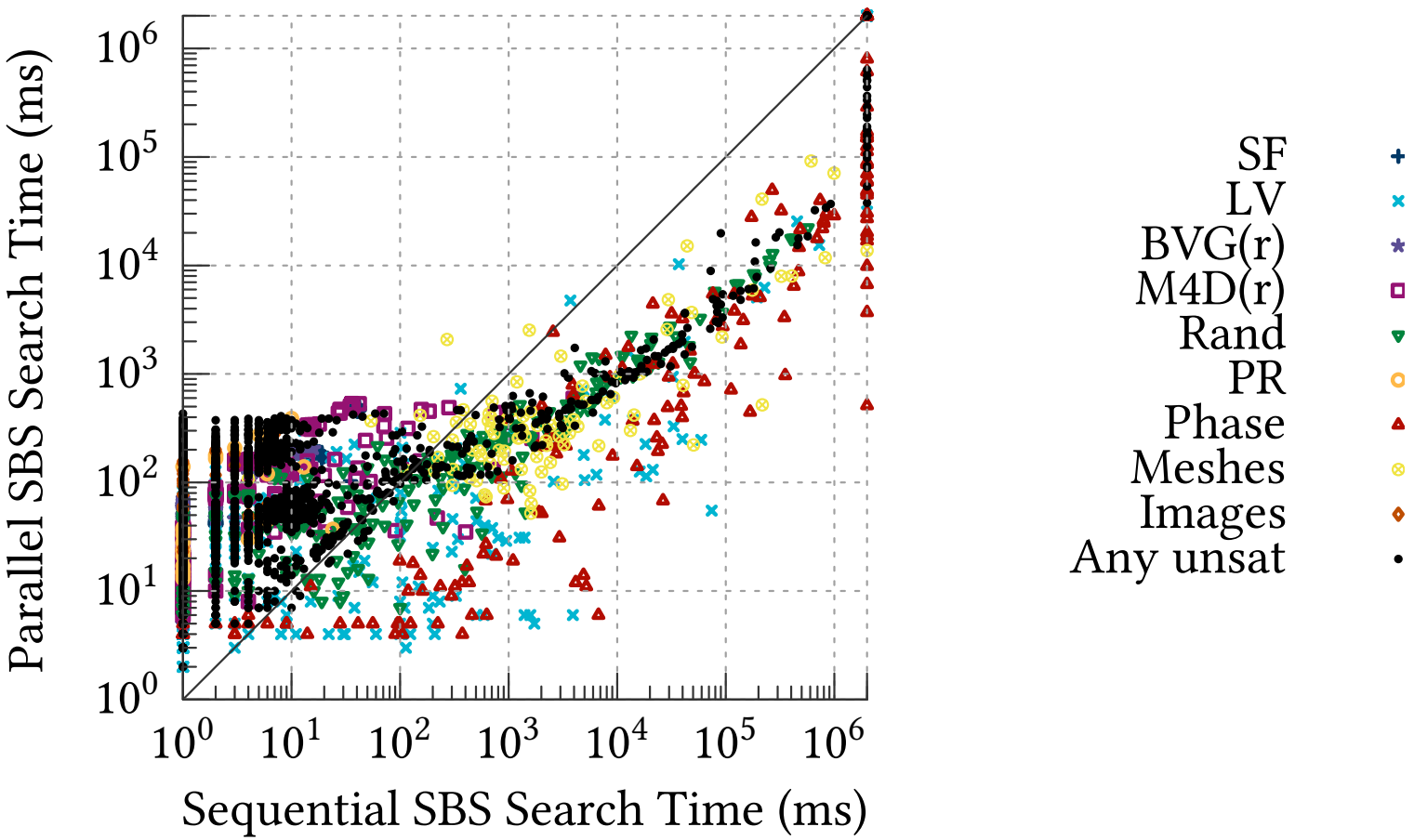
Is It Better?


2100
2000
1900
1800
1700
1600
1500
1400
Sat Instances Solved
10^2
10^3
10^4
10^5
10^6
Runtime (ms)
SBS
RRS
DDS
Degree

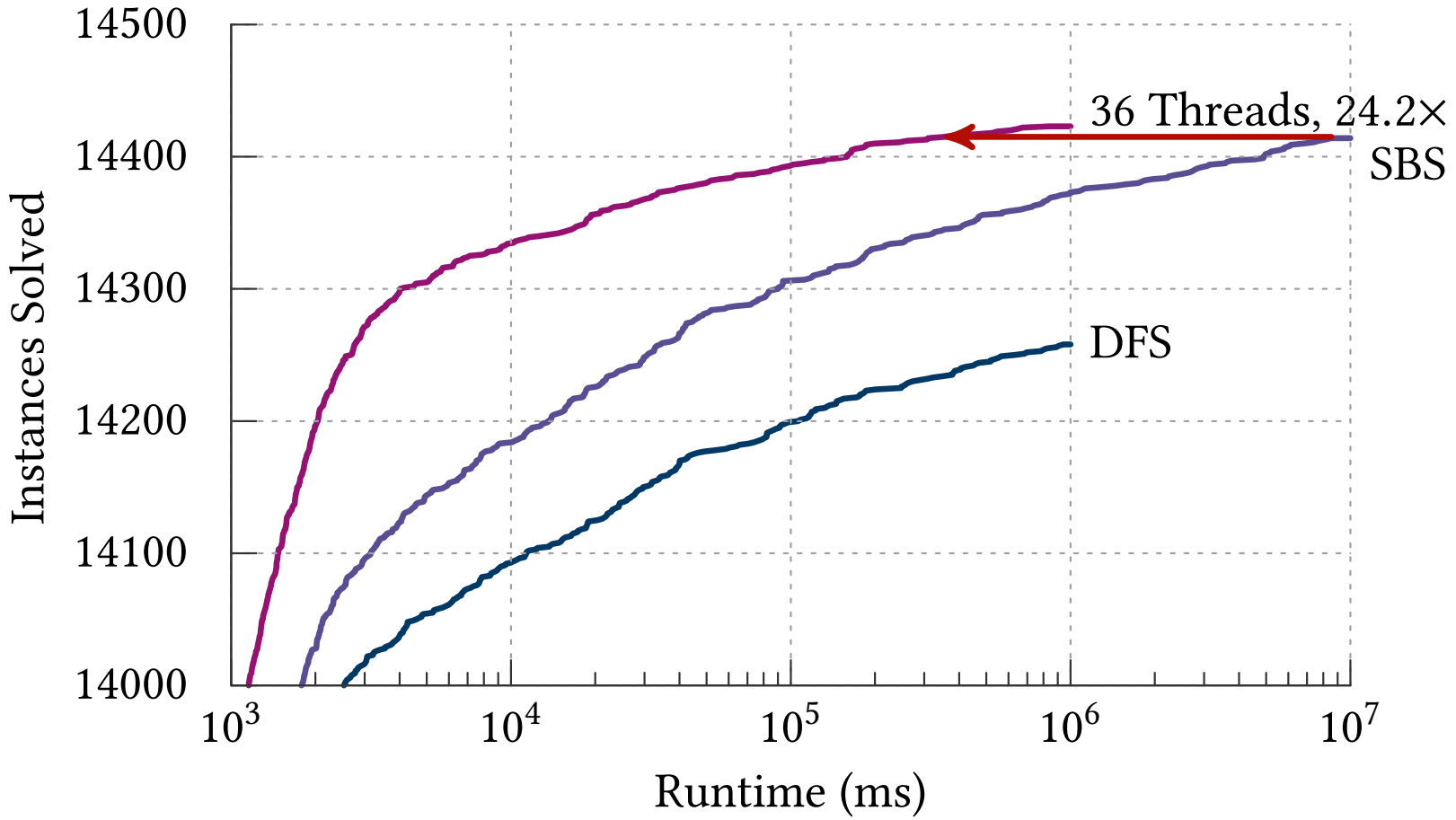
Parallel Search

- Each thread gets its own random seed.
- Barrier synchronise on restarts.
- Share nogoods.

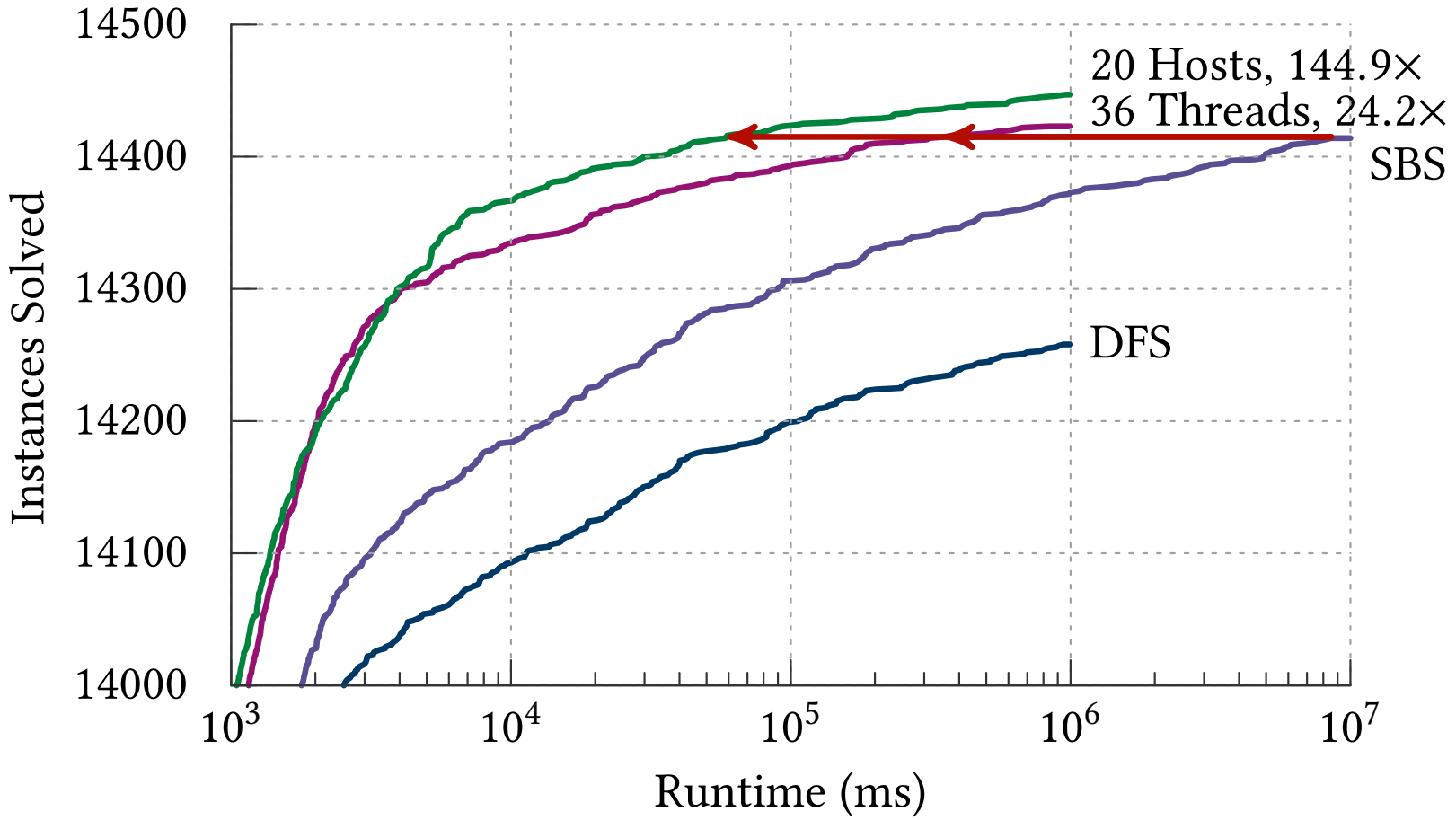
Is It Even Betterer?



Is It Even Betterer?



Is It Even Betterer?



Is There Another Way?

- This approach: assume the inputs are random graphs.
- Another possibility: “From Support Propagation to Belief Propagation in Constraint Programming.”, Gilles Pesant, JAIR 66 (2019).
- At AAAI 2020: “A Learning based Branch and Bound for Maximum Common Subgraph related Problems”, Yanli Liu, Chumin Li, Hua Jiang.

Can You Help?

- Better search
- Conflict analysis
- Symmetries
- Benchmark instances
- Side constraints
- Subgraph counting

Conclusion

- CP can help with designing other algorithms, too.
- Subgraph isomorphism is a nice testbed for learning about CP, because it's just the right amount of complicated.

http://www.dcs.gla.ac.uk/~ciaran
ciaran.mccreesh@glasgow.ac.uk

University of Glasgow