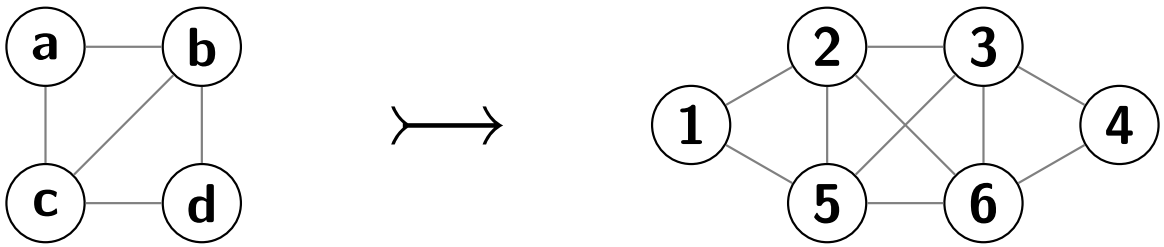
A Parallel, Backjumping Subgraph Isomorphism Algorithm using Supplemental Graphs

Ciaran McCreesh and Patrick Prosser

University of Glasgow

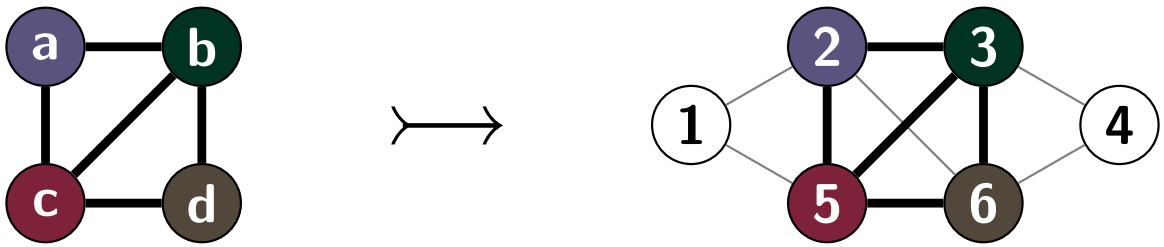
The Subgraph Isomorphism Problem

- Given a little *pattern* graph and a large *target* graph, find "a copy of" the pattern inside the target.



The Subgraph Isomorphism Problem

- Given a little *pattern* graph and a large *target* graph, find "a copy of" the pattern inside the target.



Applications

- Bioinformatics and chemistry.
- Computer vision and pattern recognition.
- Fraud detection and law enforcement.
- Model checking.
- Social network analysis.
- Code generation in compilers.
- Diseased cows.

The Basic CP Model

- One variable for each vertex in the pattern graph, with the domains being the vertices of the target graph.
- If two vertices are adjacent in the pattern, their values must be adjacent in the target.
  - In the *induced* variant, non-adjacent vertices must be mapped to non-adjacent vertices. We only discuss the non-induced variant in this talk.
- Each pattern vertex must be given a different value.
- Often enhanced: for example, we can filter based upon degree.
- Another perspective: find an injective mapping from the pattern to the target, which preserves adjacency.

Selected Previous Work

- VF2: backtracking search, interesting heuristics. Widely used.

  A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs. *Luigi P. Cordella, Pasquale Foggia, Carlo Sansone and Mario Vento*. IEEE Trans. Pattern Anal. Mach. Intell., 2004.

- LAD: locally all-different, and neighbourhood degree sequences.

  AllDifferent-based filtering for subgraph isomorphism. *Christine Solnon*. Artif. Intell., 2010.

- SND: distance-based filtering.

  Scoring-Based Neighborhood Dominance for the Subgraph Isomorphism Problem. *Gilles Audemard, Christophe Lecoutre, Mouny Samy Modeliar, Gilles Goncalves and Daniel Porumbel*. CP 2014.

Being Clever is Expensive

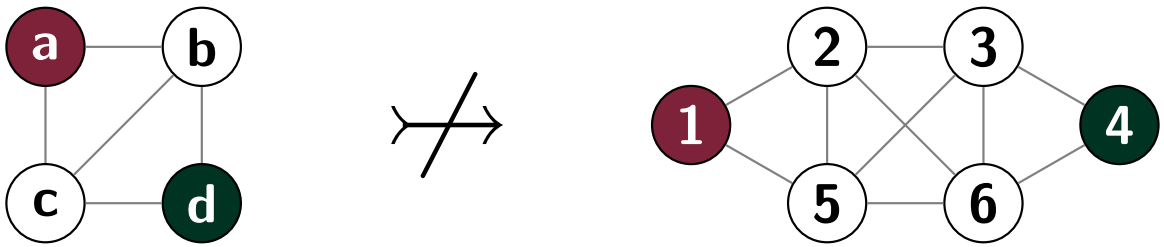
- We want to work with up to 1,000 pattern vertices, and 10,000 target vertices.
- When LAD and SND fail, they often manage less than one recursive call per second, particularly with larger target graphs.

Our Approach

- Cheaper inference: 10^4 to 10^6 recursive calls per second.
- Expensive preprocessing once at the top of search, rather than computing distances and degree sequences during search.
- FC-CBJ instead of MAC.
- A counting-based all-different propagator, which does more than pairwise-$\neq$ AC but less than GAC.
- Bit-parallelism for all major operations.
- Thread-parallel search.

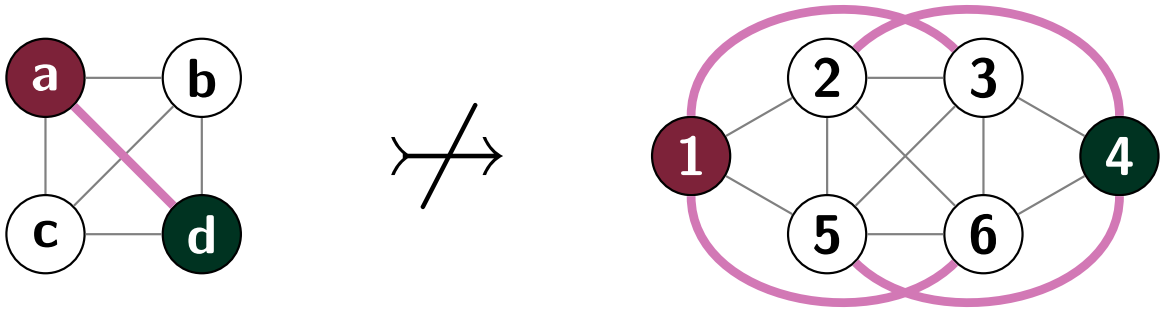
Supplemental Graphs

- Adjacent vertices must be mapped to adjacent vertices.
- Used by SND:
  - Vertices that are distance 2 apart must be mapped to vertices that are within distance 2.
  - Vertices that are distance k apart must be mapped to vertices that are within distance k.



Supplemental Graphs

- G^d is the graph with the same vertex set as G, and an edge between v and w if the distance between v and w in G is at most d.
- For any d, a subgraph isomorphism $i : P \rightarrowtail T$ is also a subgraph isomorphism $i^d : P^d \rightarrowtail T^d$.



Supplemental Graphs

- We can do something stronger: rather than looking at distances, we can look at (simple) paths, and we can count how many there are.
- This is NP-hard in general, but only lengths 2 and 3 and counts of 2 and 3 are useful in practice.
- We construct these graph pairs once, at the top of search, and use them for degree-based filtering at the top of search, and "adjacency" filtering during search.

Counting-Based All-Different

- One pass through the variables, from smallest domain to largest.
- Track unions of domains as we go along.
- Eliminate any detected Hall sets from future variables.
- This can miss some deletions, but has a very fast bitset implementation for large domains.

Backjumping

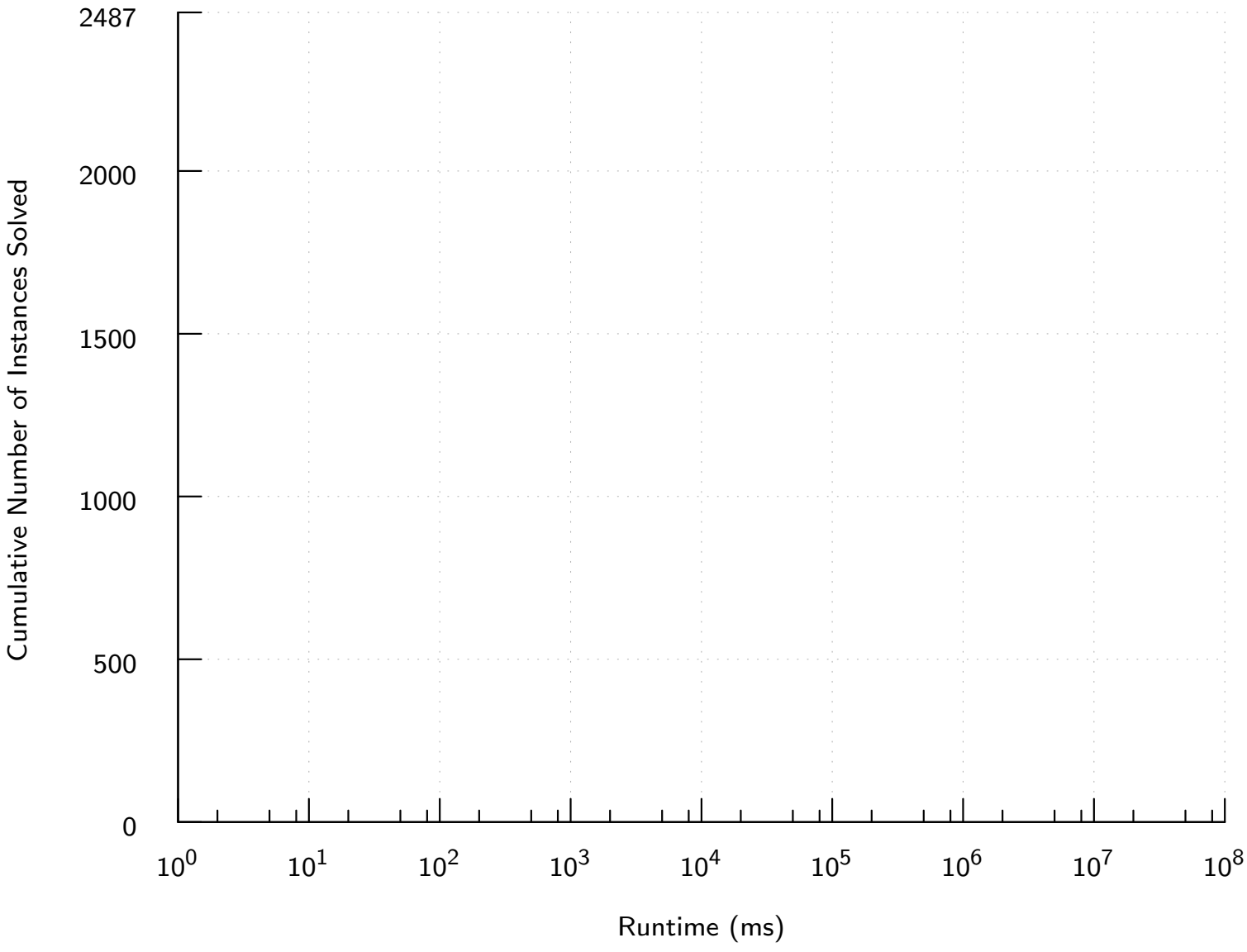
- Rather than backtracking on failure, we can sometimes show it's safe to jump back several levels immediately.
- This can be done with without explicit conflict sets, and without modifying propagators.
- Producing smaller conflict sets from a failed all-different gives better results.

Experimental Setup

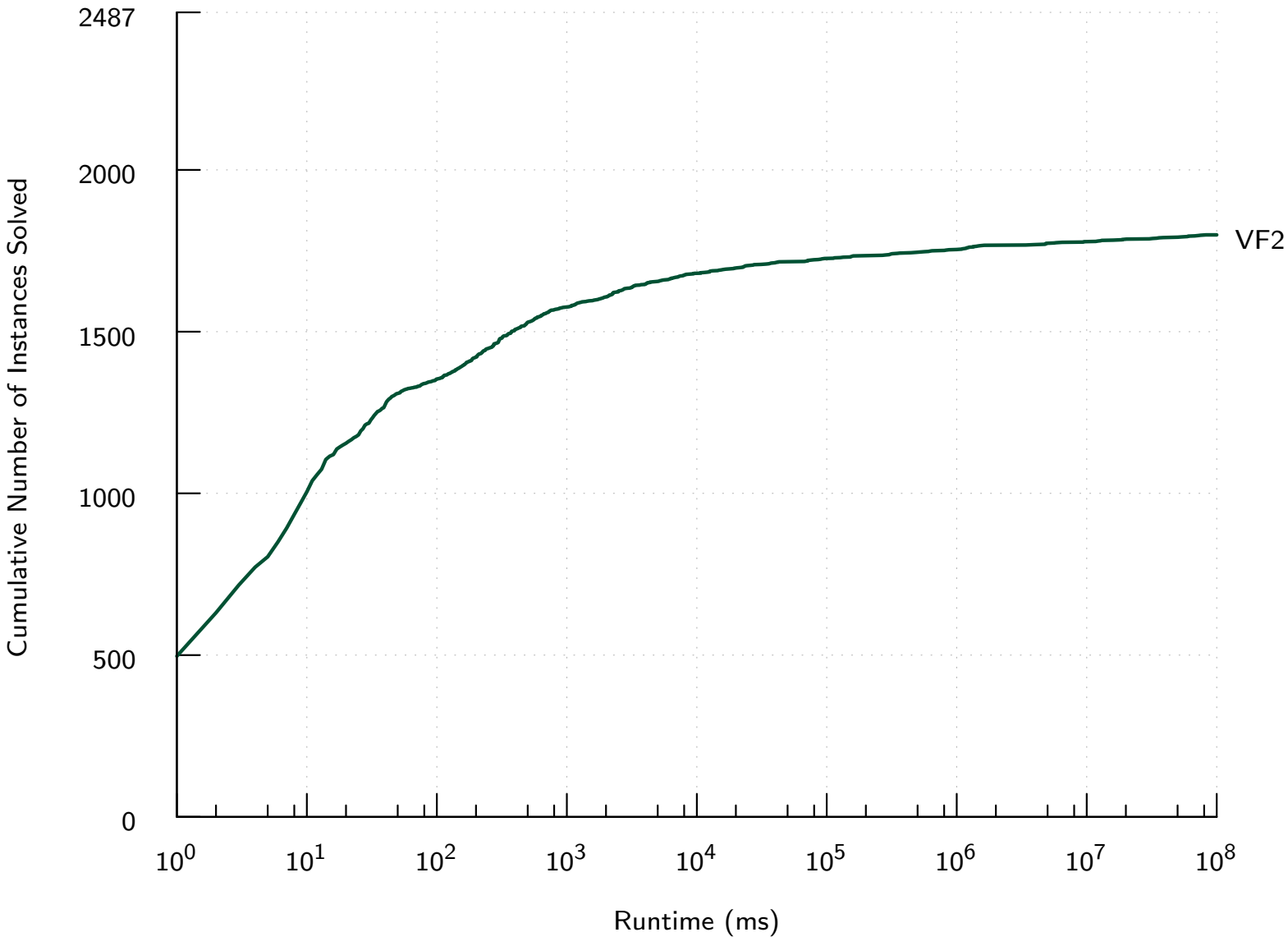
- Dual Intel Xeon E5-2640 v2 (Q3'13), 64GBytes RAM. Our code is C++, LAD and VFLib are C, SND is Java.
- 2,487 pairs of instances, selected by other people:
  - Real-world graphs.
  - 2D images and 3D meshes from computer vision applications.
  - Random (simple, scale-free, 4D mesh, bounded degree).
- Up to 900 vertices and 12,410 edges in the pattern, and 5,944 vertices and 34,210 edges in a target. A mix of satisfiable and unsatisfiable instances, although there is a bias towards satisfiable instances...
- Please donate your subgraph isomorphism problem instances:

  `http://liris.cnrs.fr/csolnon/SIP.html`

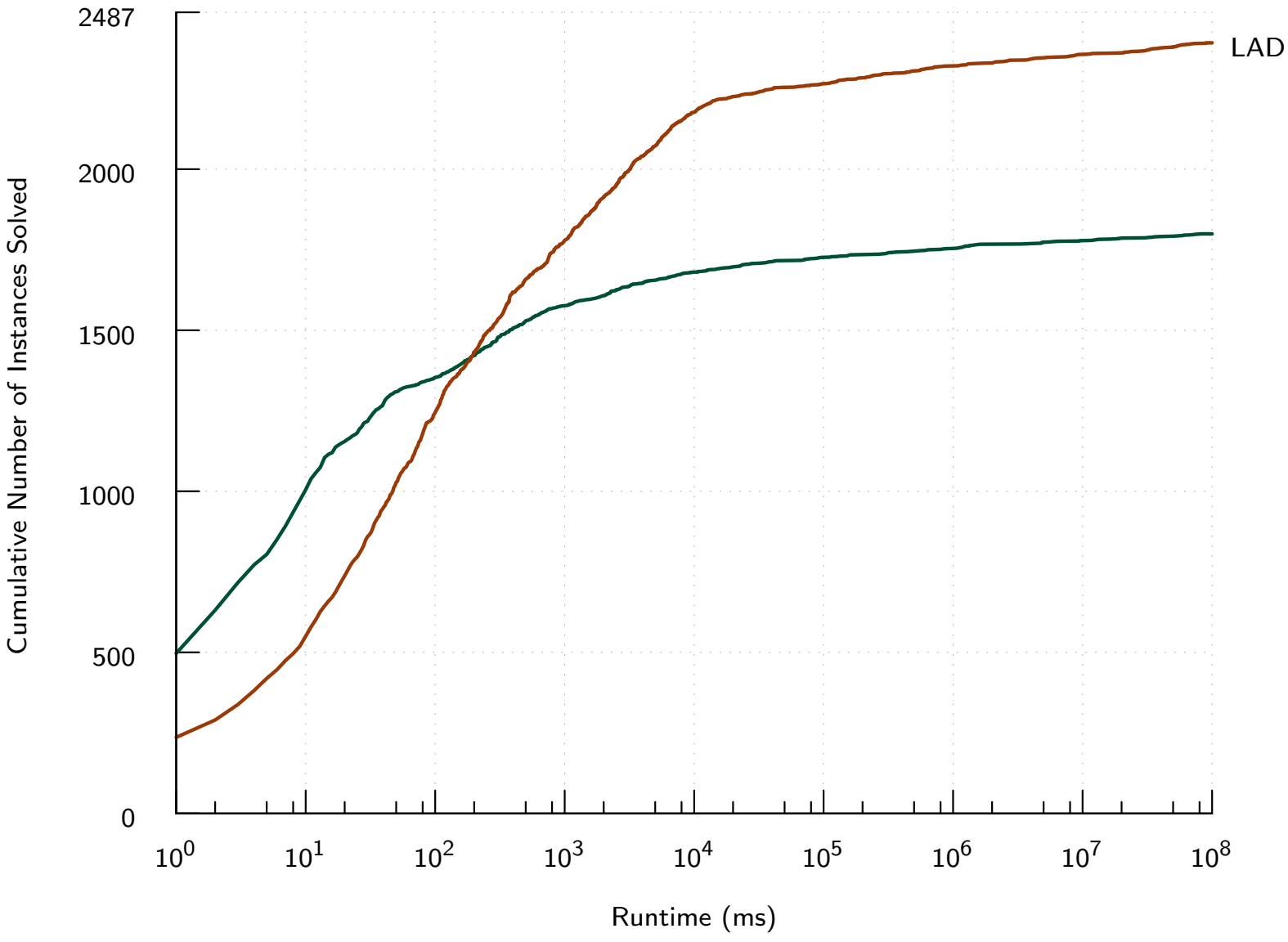
Cumulative Instances Solved



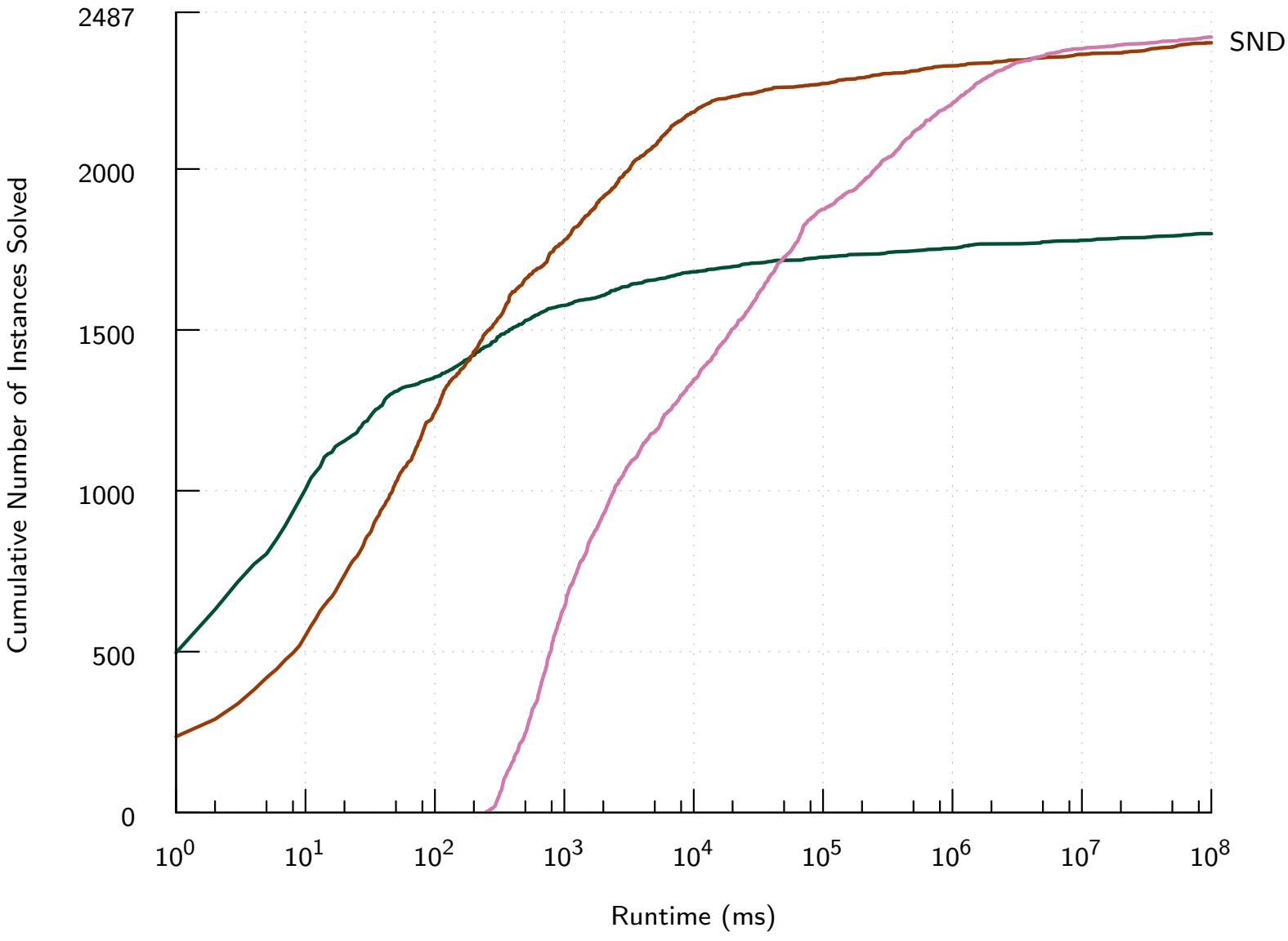
Cumulative Instances Solved



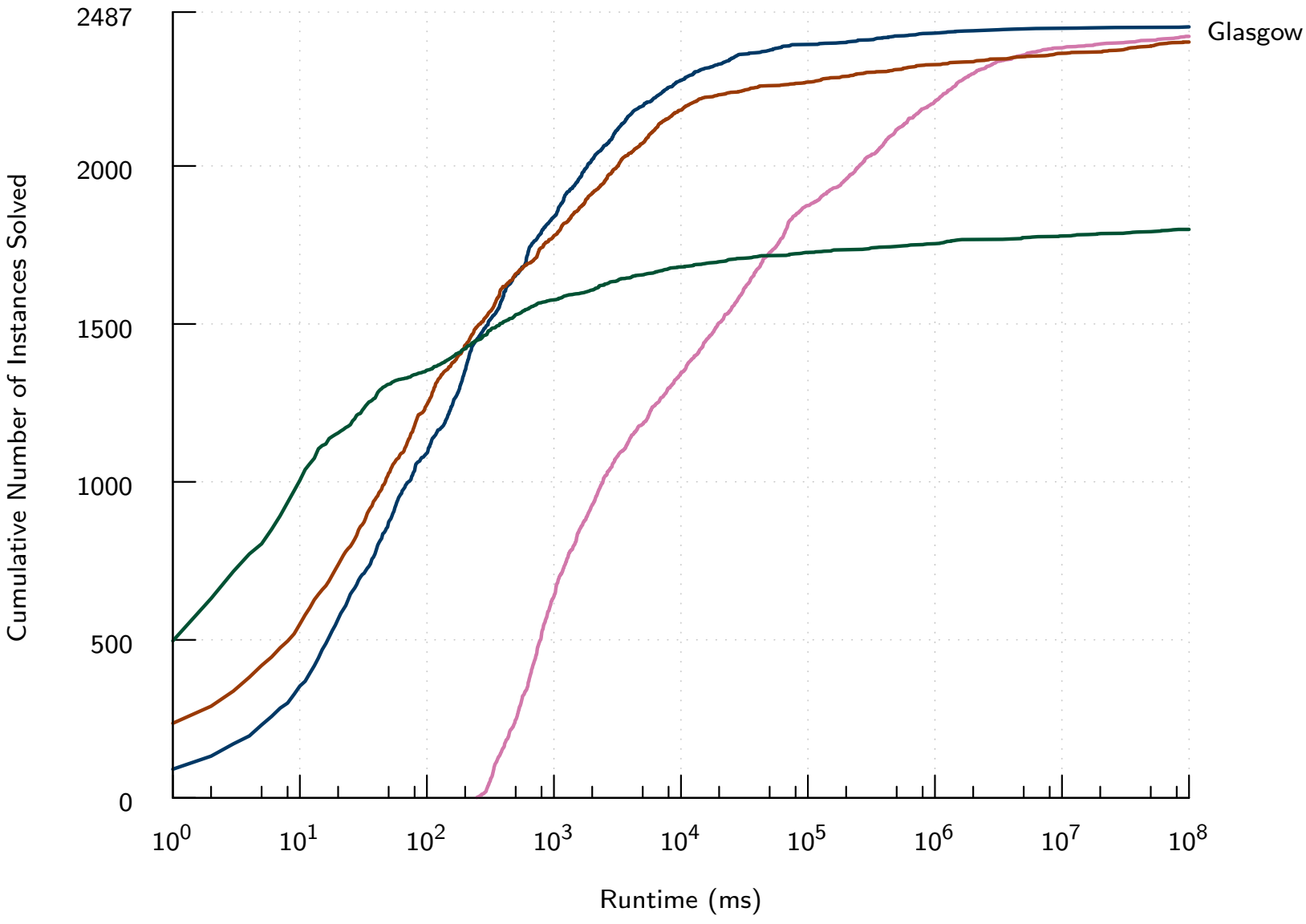
Cumulative Instances Solved



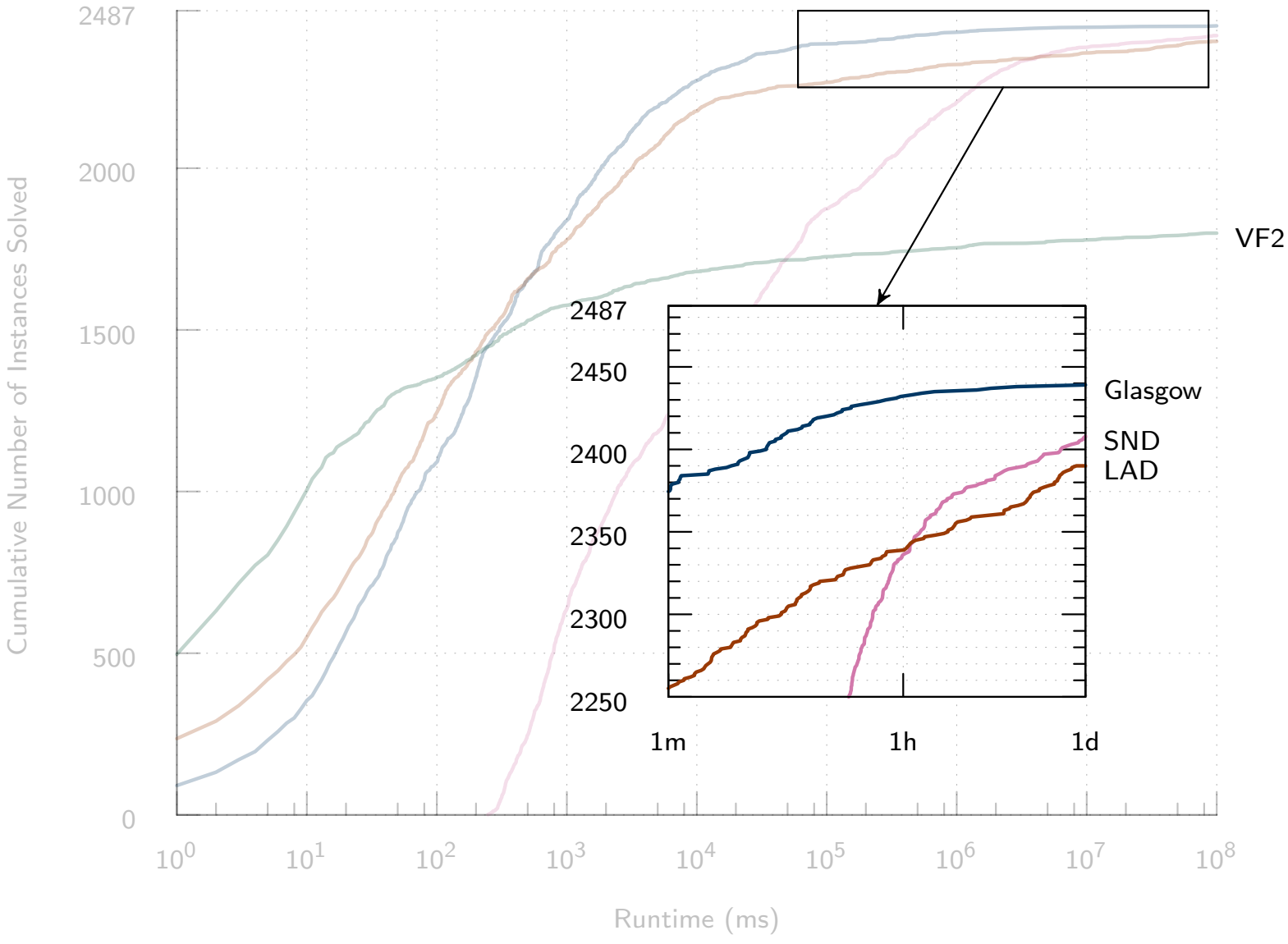
Cumulative Instances Solved



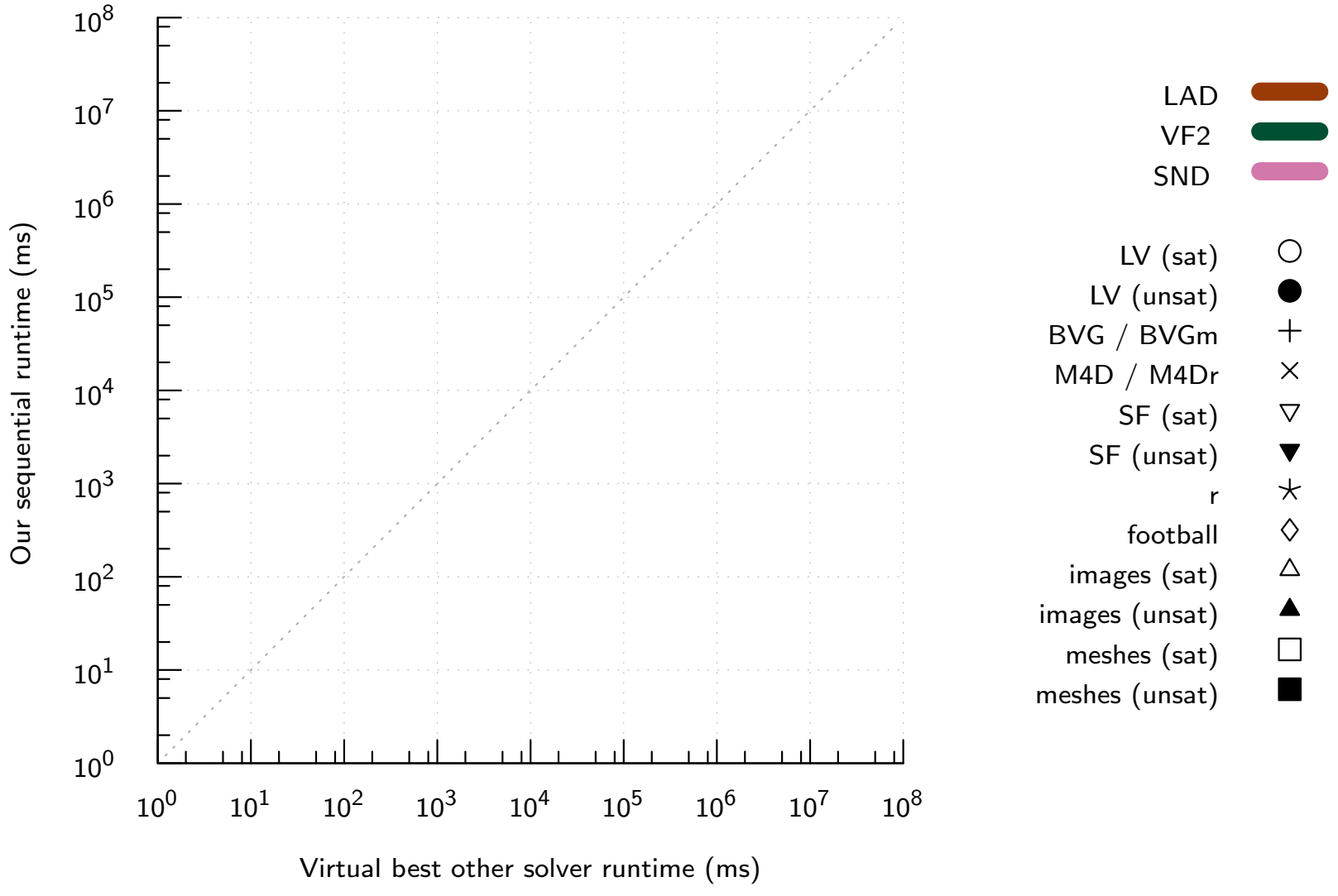
Cumulative Instances Solved



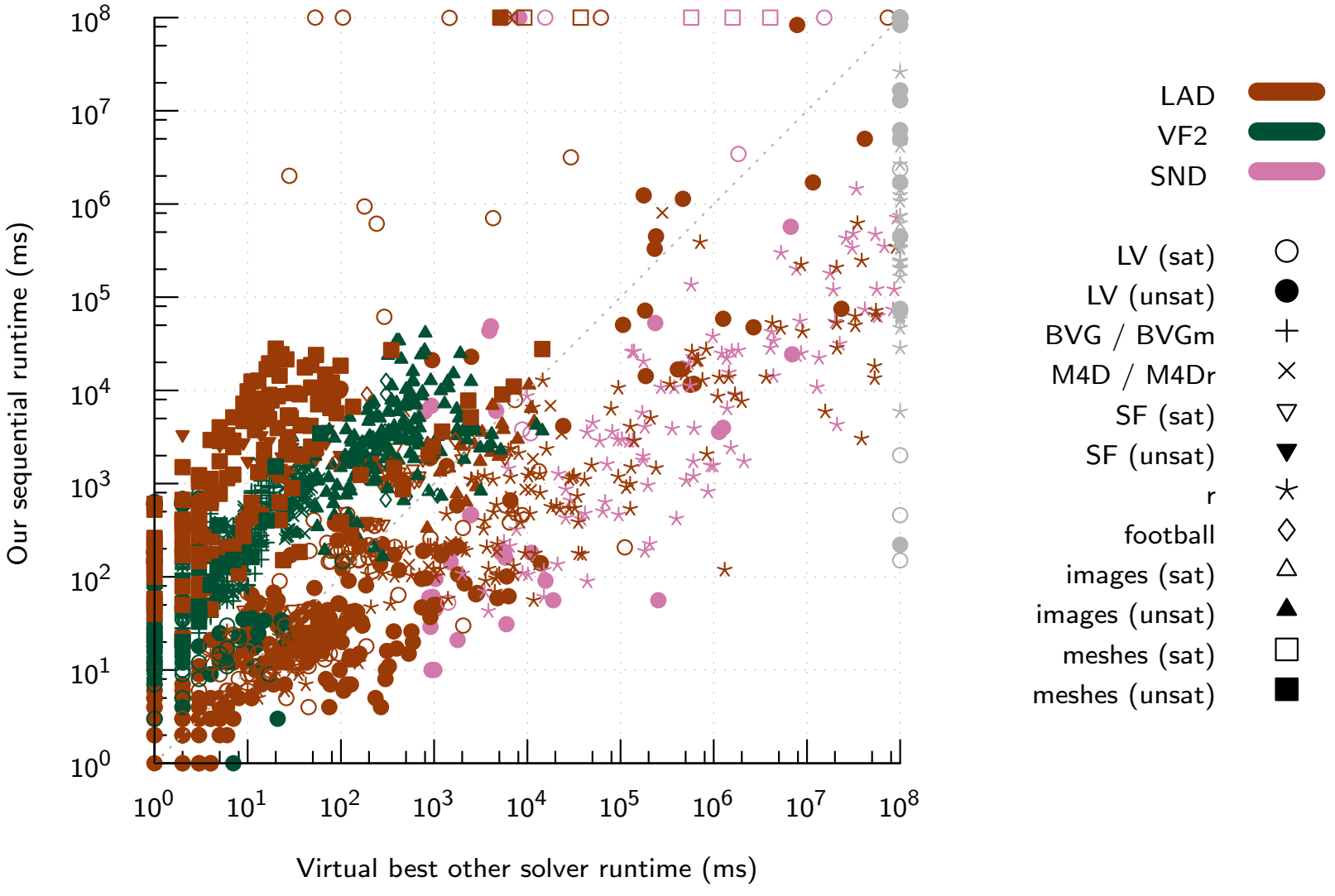
Cumulative Instances Solved



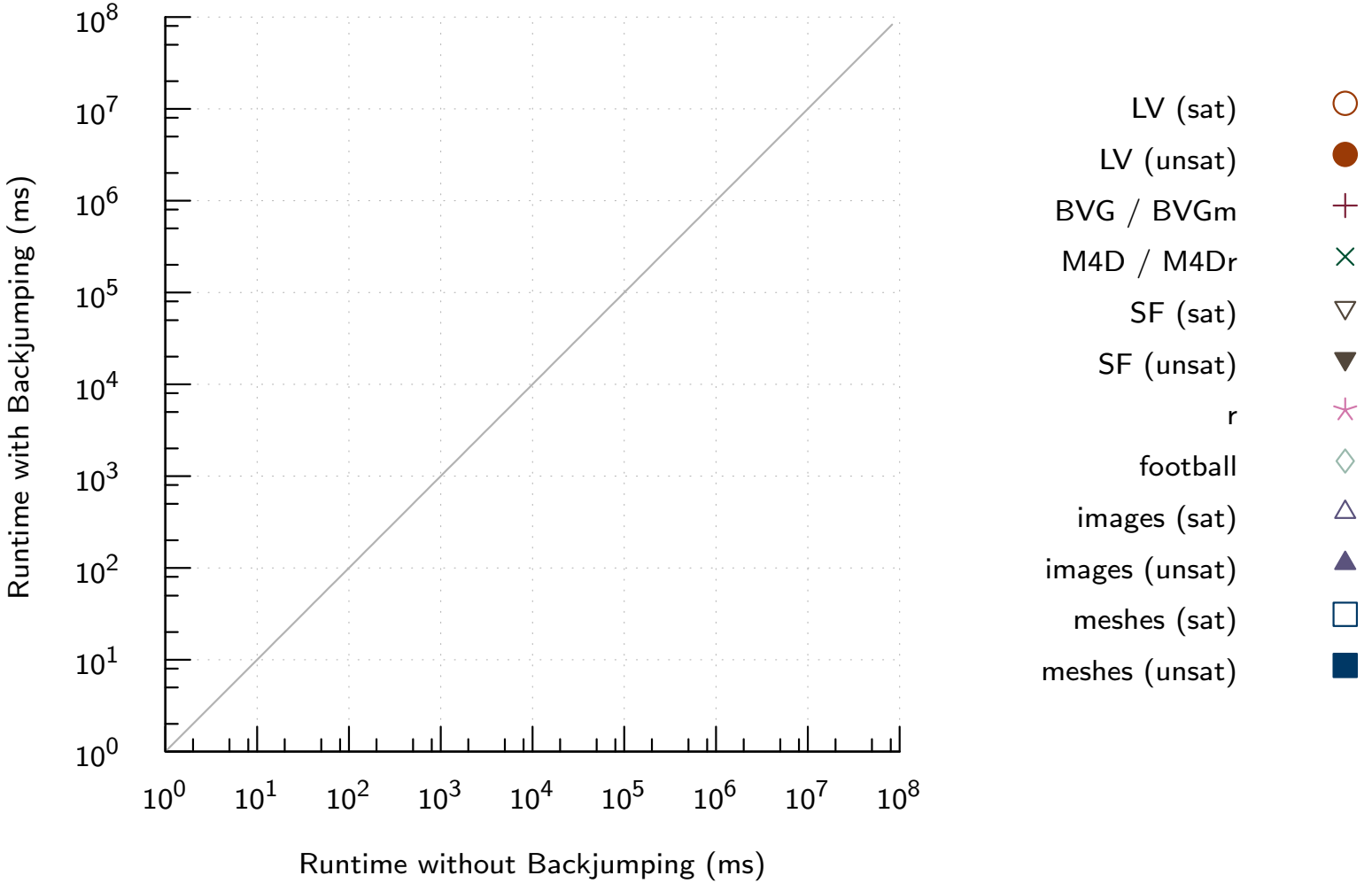
Versus Virtual Best Other Solver



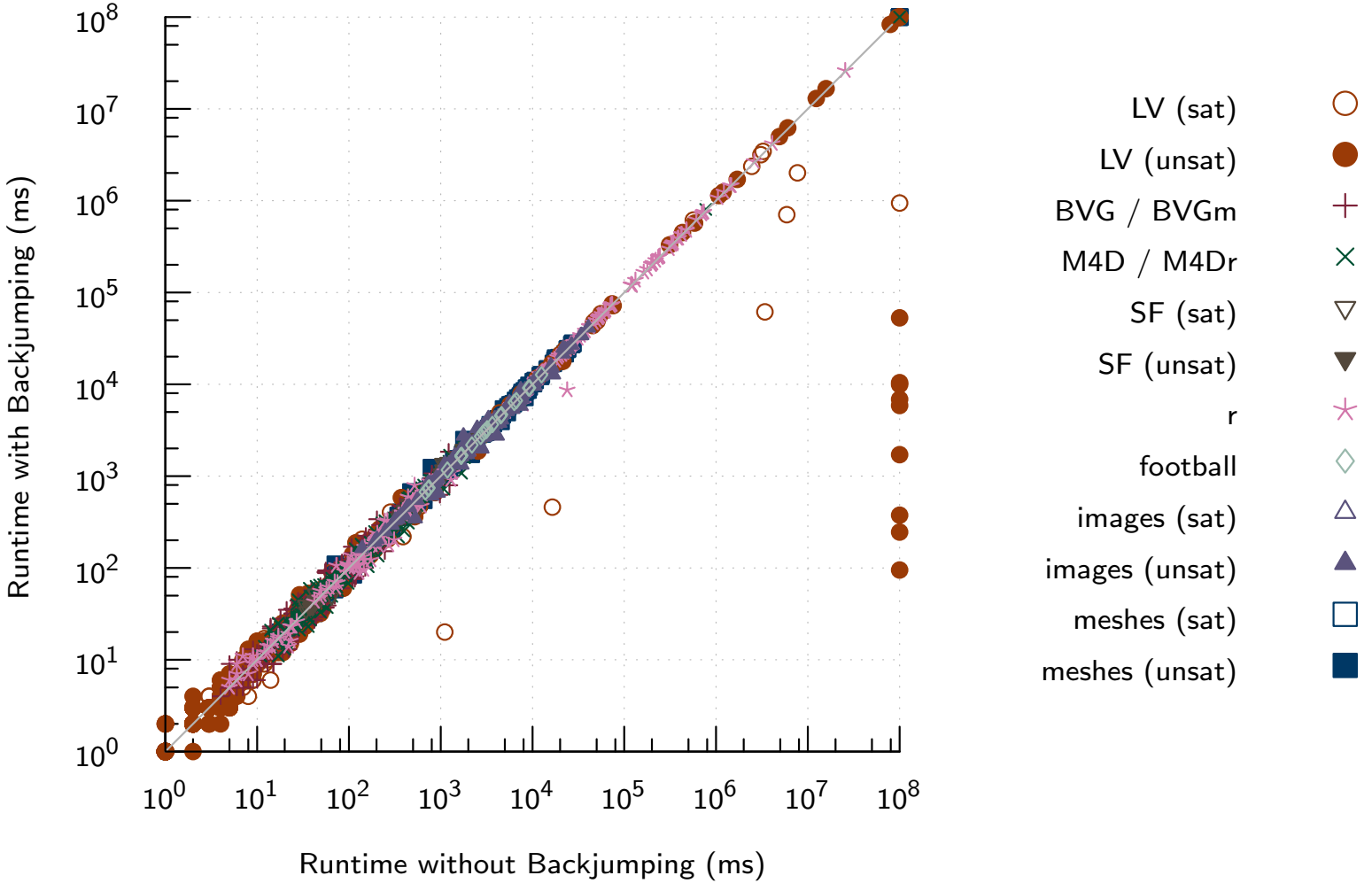
Versus Virtual Best Other Solver



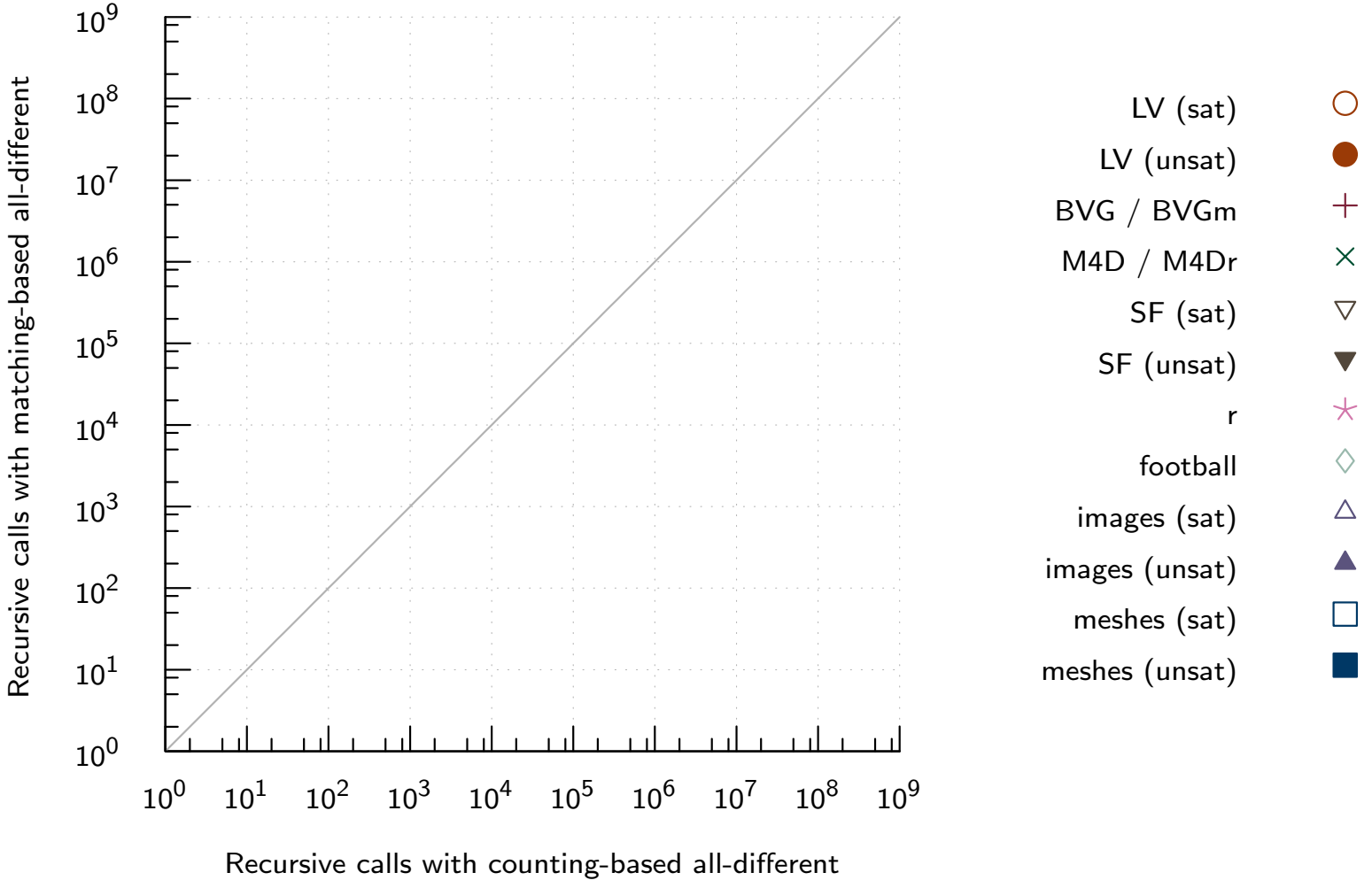
Backjumping?



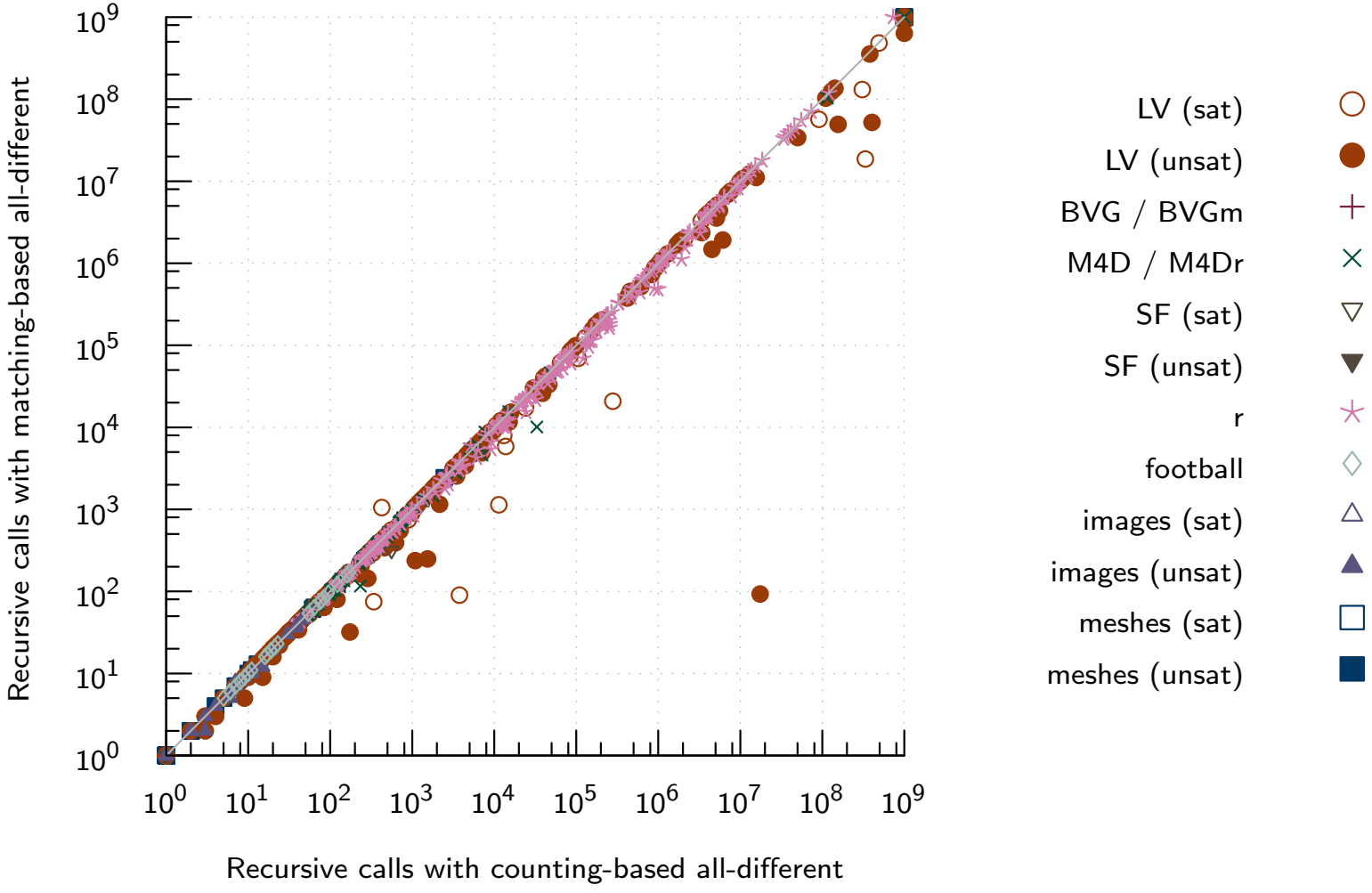
Backjumping?



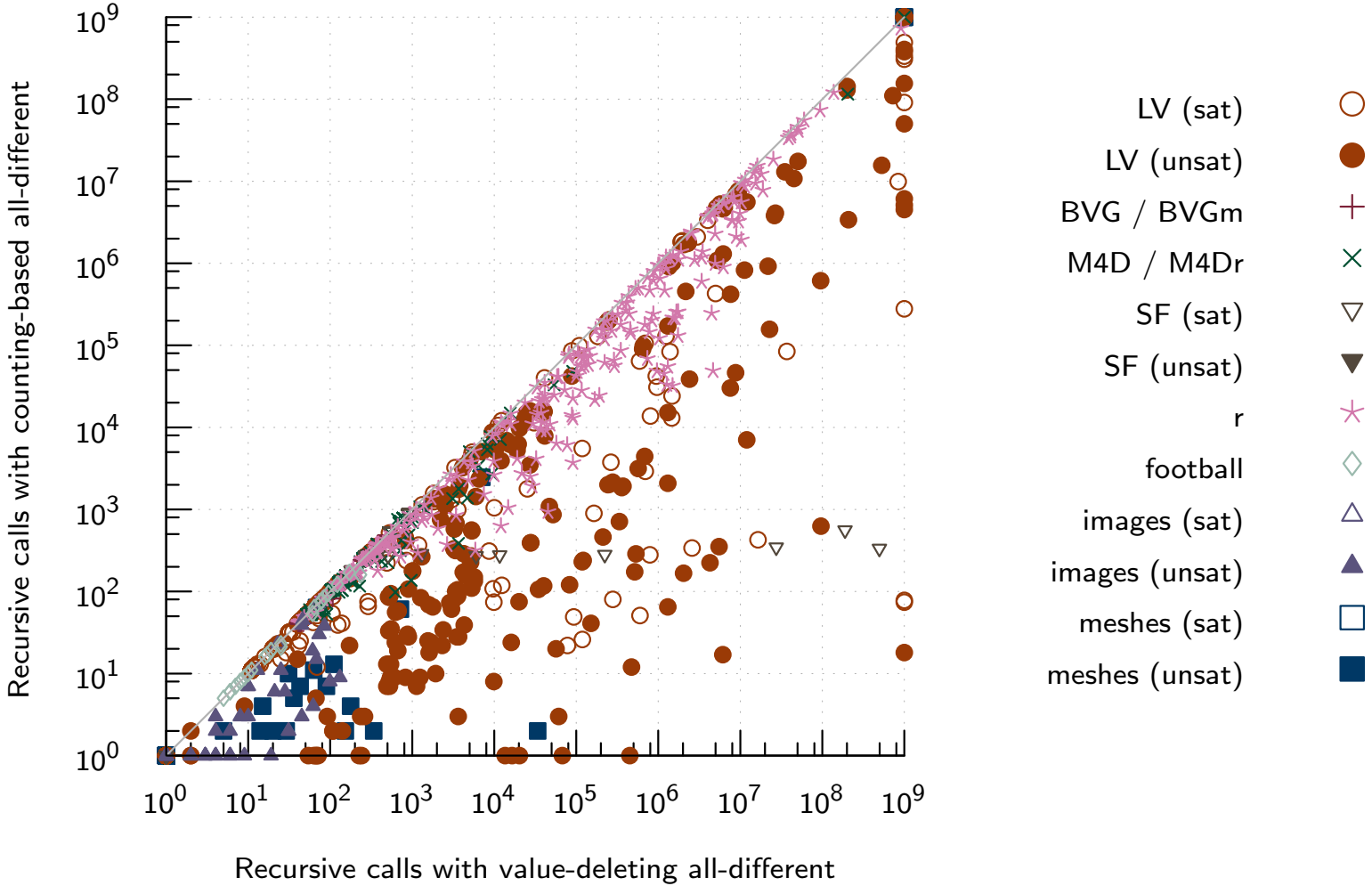
Counting All-Different?



Counting All-Different?



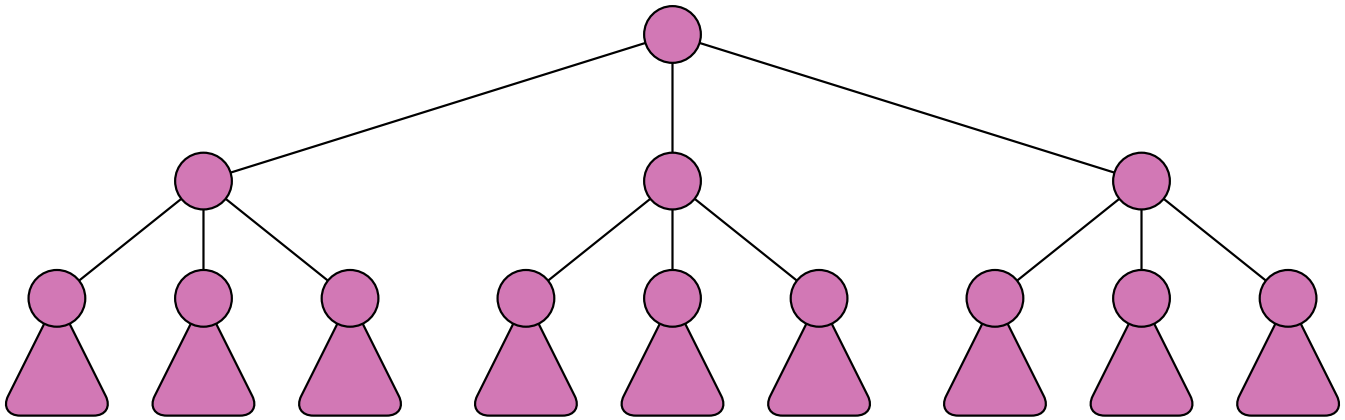
Counting All-Different?



Parallel Preprocessing

- Constructing supplemental graphs is expensive, but we have lots of cores.
- Parallelising the loops is entirely (mostly...) routine and not very interesting, but good at offsetting the cost of supplemental graph construction.

Parallel Tree-Search



Parallel Tree-Search

- Explicit deterministic early-first work stealing, to offset incorrect decisions early in search.
  - There are EHPs, and this gets rid of them more cheaply than discrepancy searches.
- Backjumping is a nested parallel fold with left-zero elements.
- Speculative, so we should not expect linear speedups (even on unsat instances).

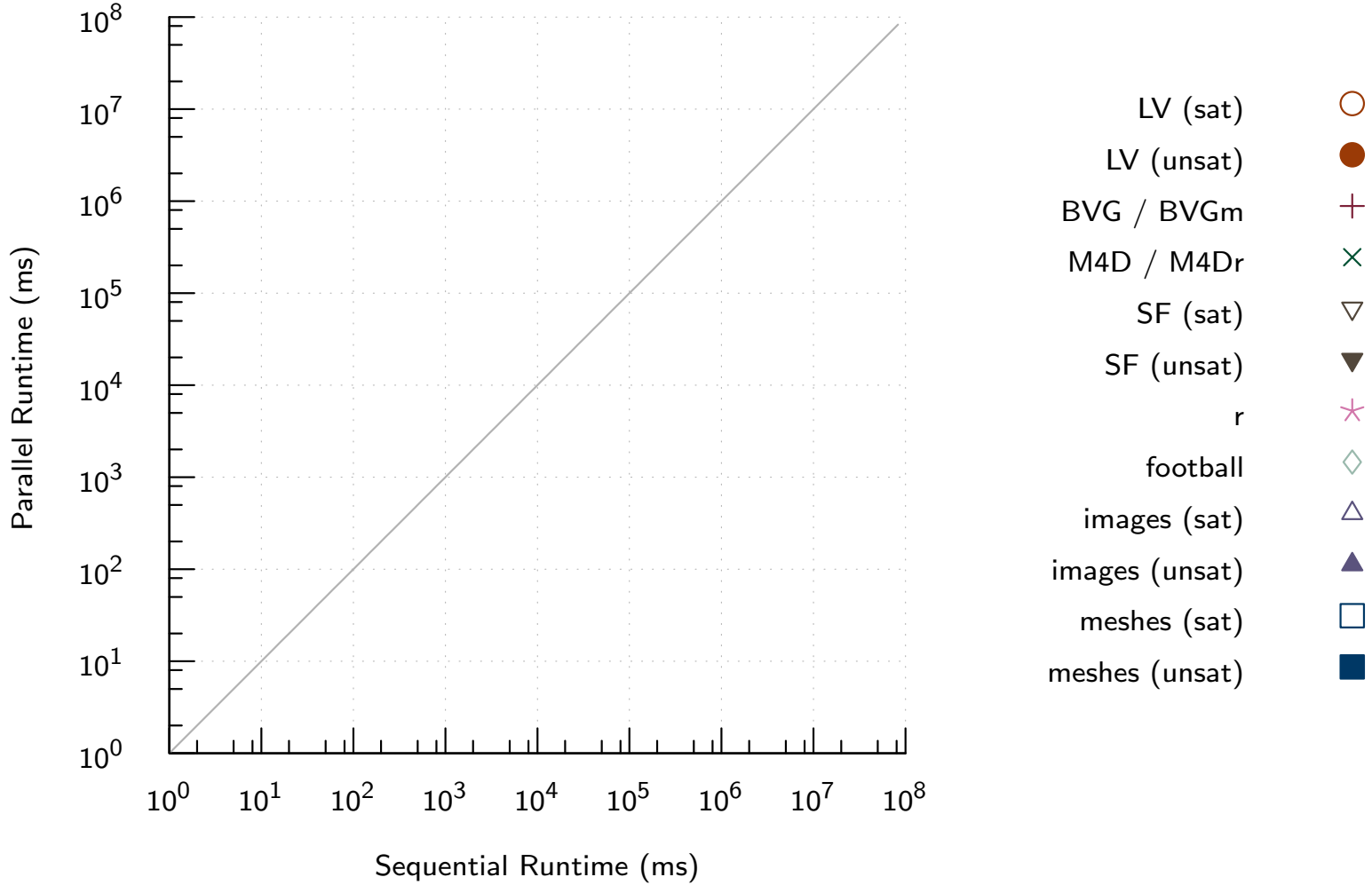
Sidenote on Safety, Parallelism, and Benchmarking

- All comparisons are against a dedicated sequential implementation of a strong algorithm, not a threaded implementation run with one thread.
- The way we do parallelism guarantees:
  - Parallel search will never be substantially worse than sequential search.
  - Adding more cores will never make things substantially worse (excluding hardware weirdness).
  - Running it twice will give more or less the same runtimes.

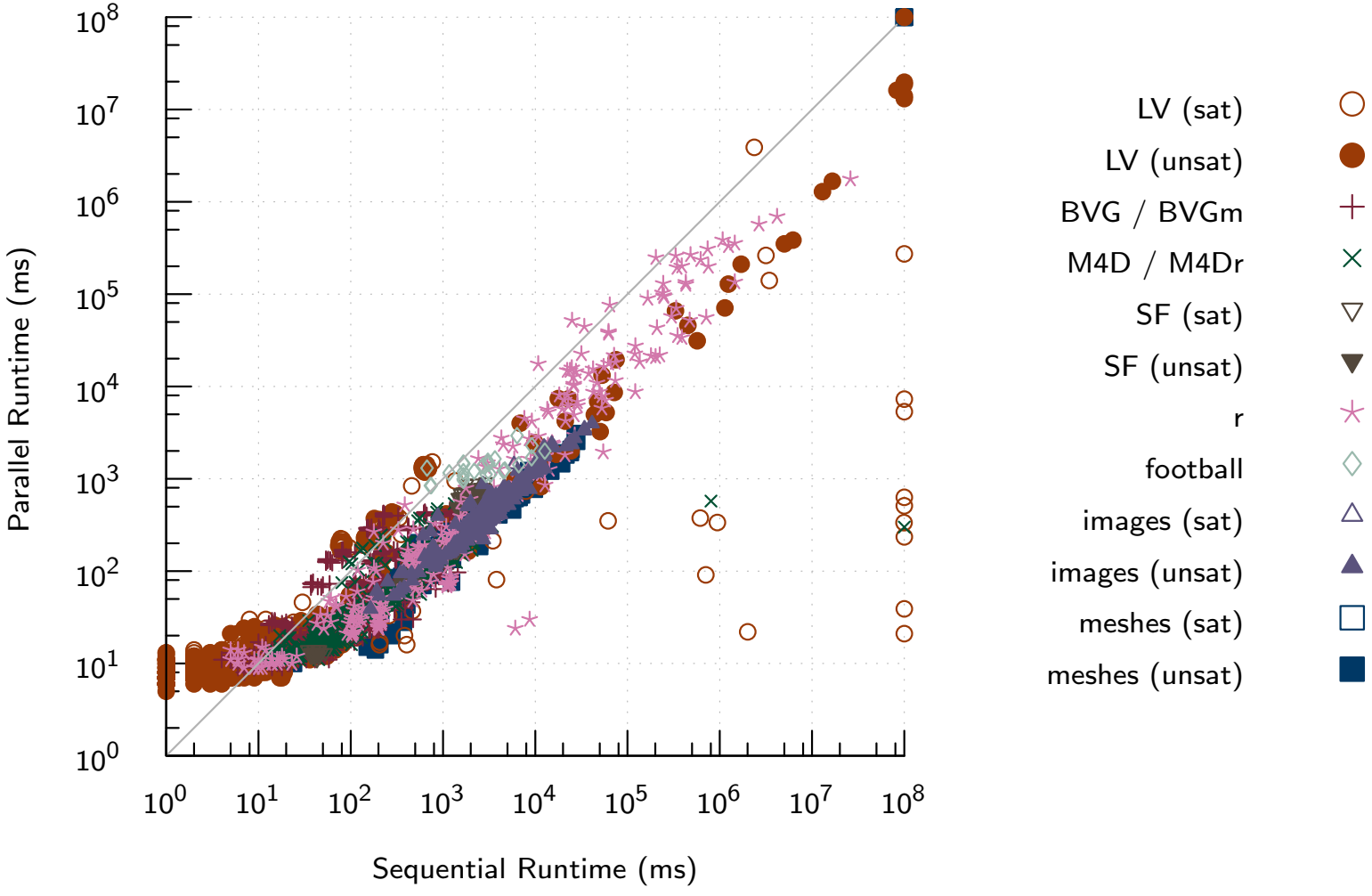
Parallel Experiments

- 2 CPUs, 8 cores per CPU, hyper-threaded, so 32 software threads.
- We do *not* have 32 times the computation power, and only about twice the memory bandwidth...

Speedups


Parallel Runtime (ms)
Sequential Runtime (ms)
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7
10^8
LV (sat)
LV (unsat)
BVG / BVGm
M4D / M4Dr
SF (sat)
SF (unsat)
r
football
images (sat)
images (unsat)
meshes (sat)
meshes (unsat)

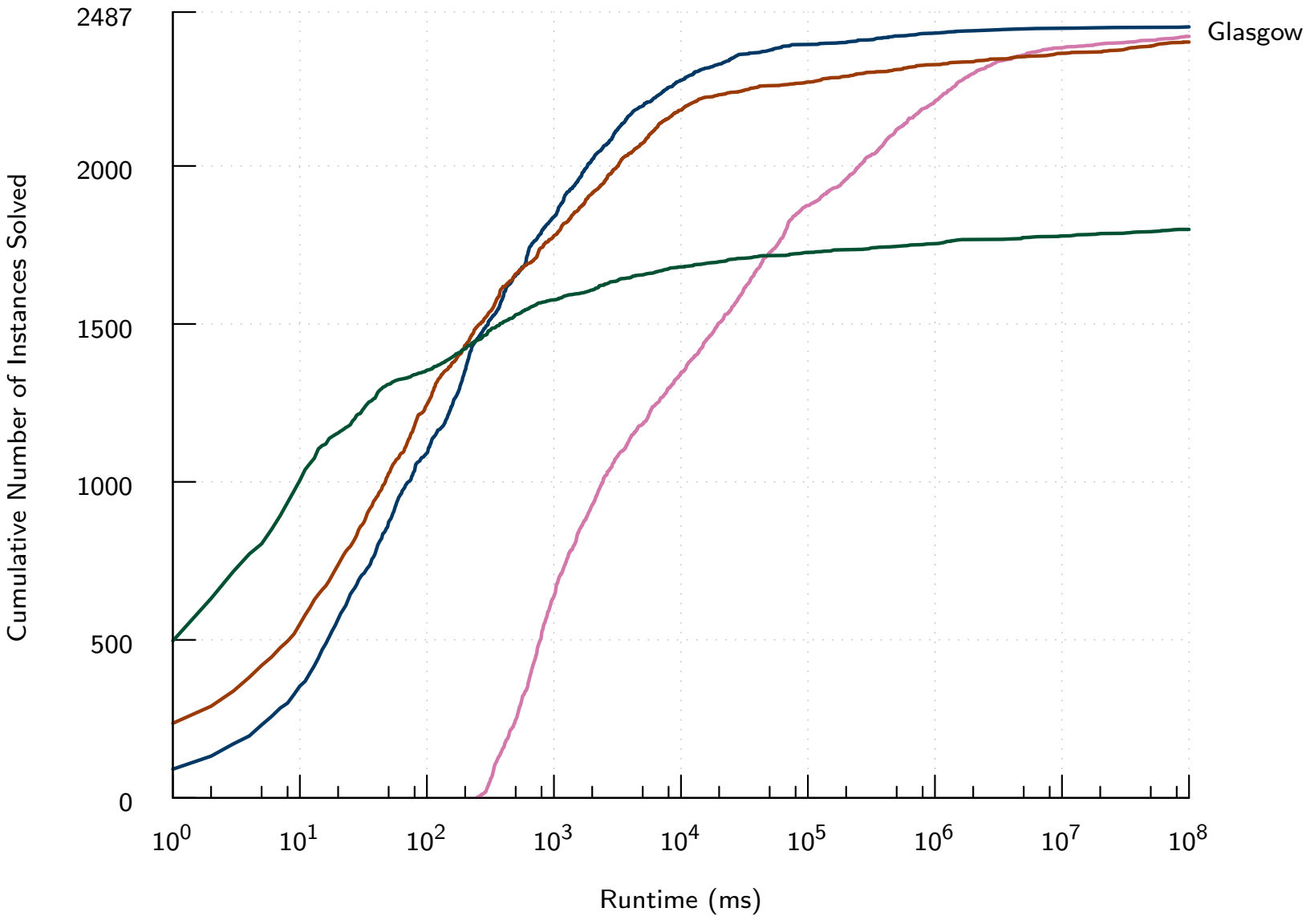
Speedups


Parallel Runtime (ms)
Sequential Runtime (ms)
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7
10^8
LV (sat)
LV (unsat)
BVG / BVGm
M4D / M4Dr
SF (sat)
SF (unsat)
r
football
images (sat)
images (unsat)
meshes (sat)
meshes (unsat)

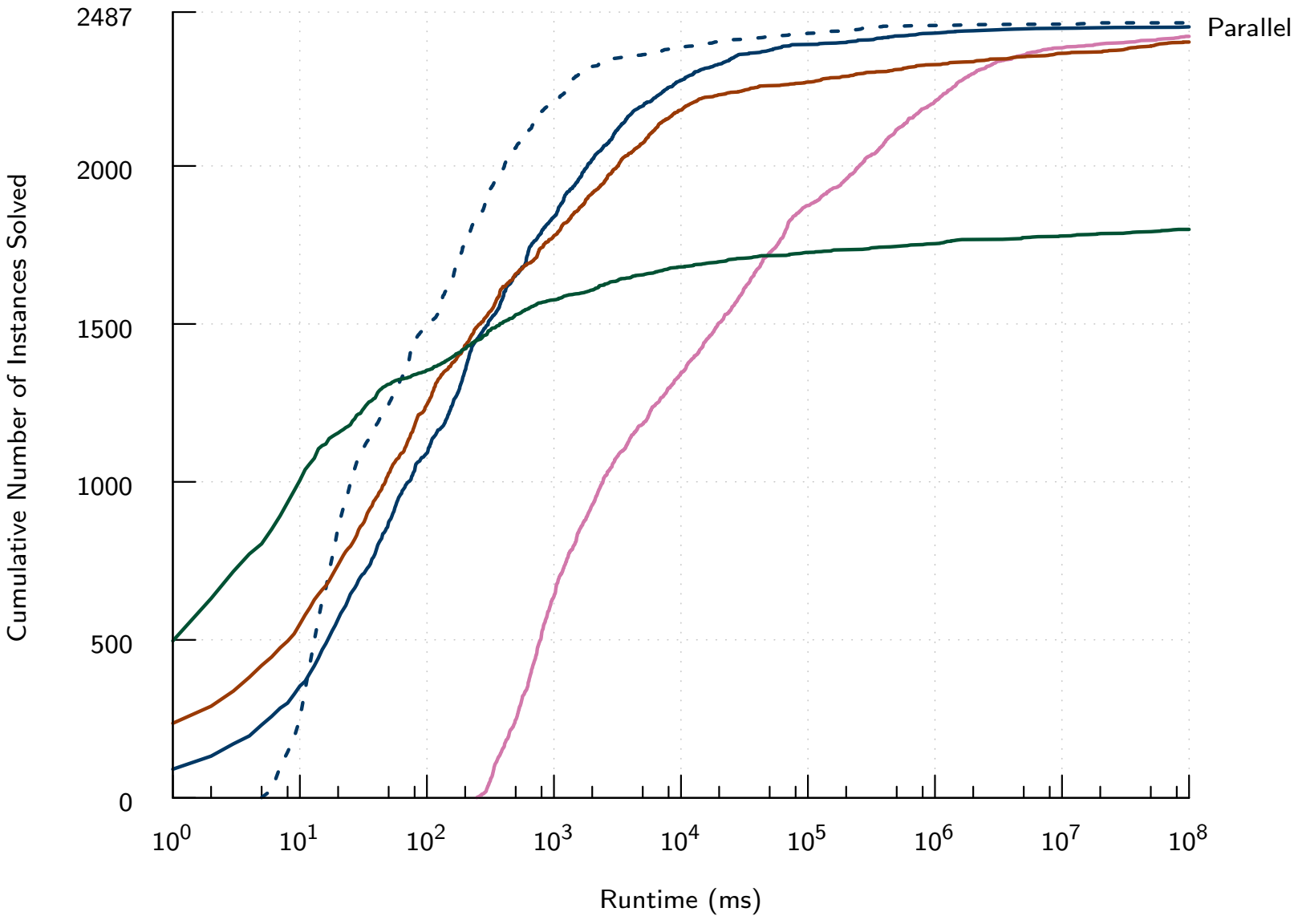
Is Comparing Sequential and Parallel Solvers Fair?

- Personal view: single-threaded for understanding "inside search" behaviour, but "horse race" benchmarks should be a free-for-all.
- You're paying for multi-core even if you aren't using it.
- Java uses multiple cores even for sequential code, so the sequential benchmarks were unfair.
- You are welcome to disagree, and ignore the following two slides.

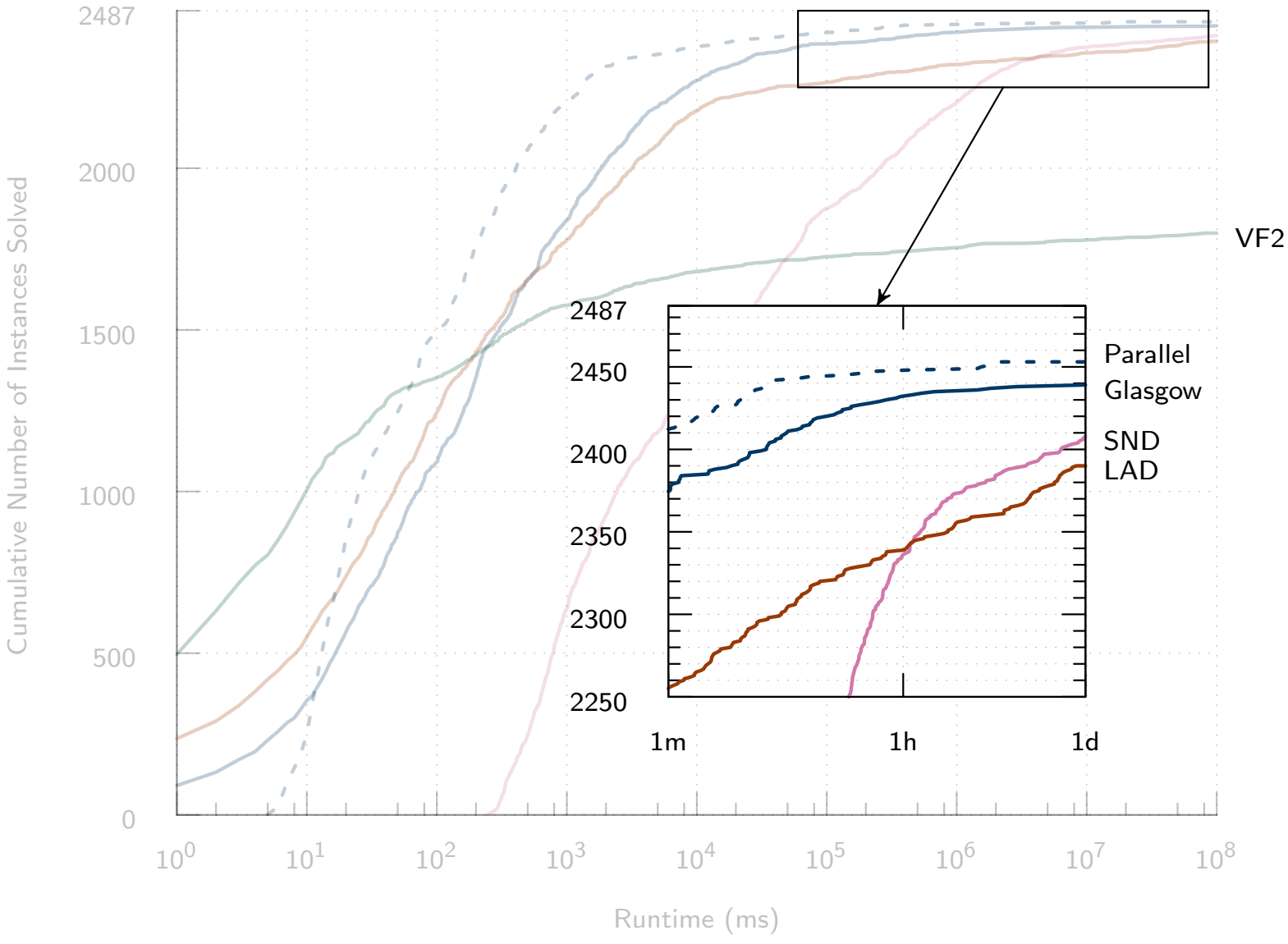
Cumulative Instances Solved



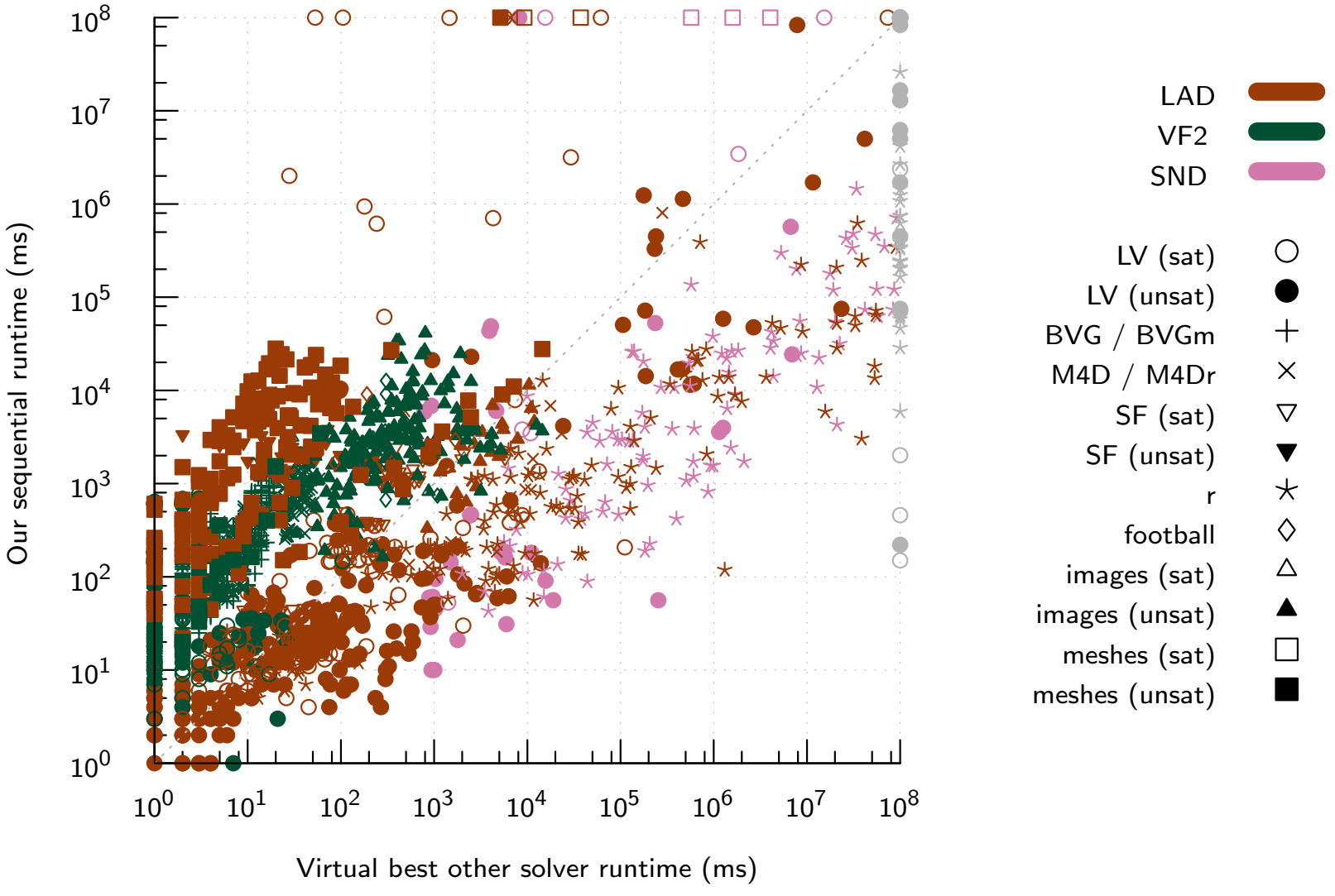
Cumulative Instances Solved



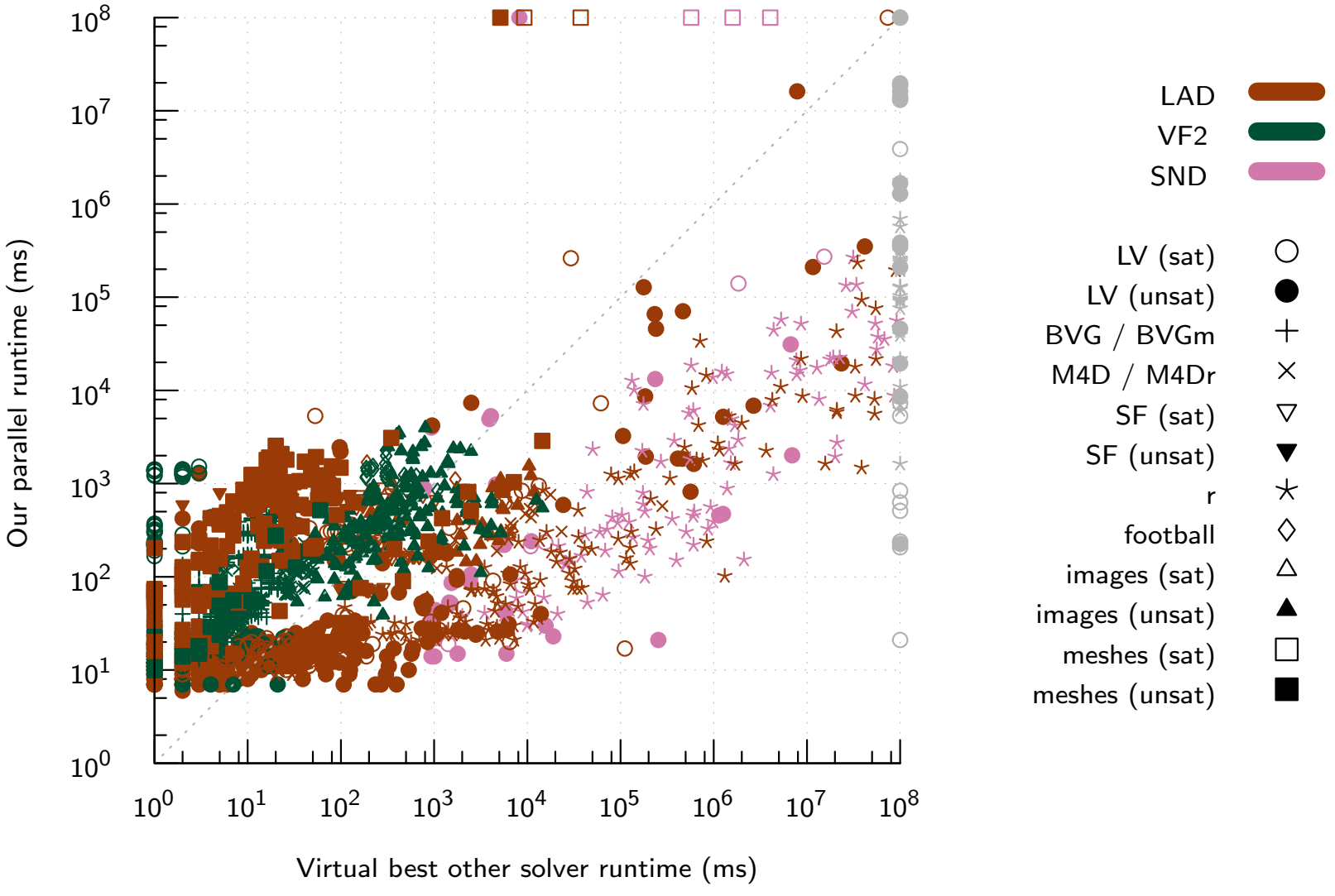
Cumulative Instances Solved



Versus Virtual Best Other Solver



Versus Virtual Best Other Solver



Conspicuously Absent From This Talk

- Labels? Directed edges? Induced? Wildcards?
- Why that choice of supplemental graphs, using only paths of lengths 2 and 3? It seems rather arbitrary. Can we do better with domain knowledge, or with portfolios?
- Caching supplemental targets (graph databases) or patterns (code generation, kidney exchange).
- Why does that counting all-different propagator do so well in practice? Is it only subgraph isomorphism where it helps?

University of Glasgow

Reproduce, replicate, or recompute this paper:
`http://dcs.gla.ac.uk/~ciaran`

Tell me whether it worked:
`c.mccreesh.1@research.gla.ac.uk`

Donate your subgraph isomorphism problem instances:
`http://liris.cnrs.fr/csolnon/SIP.html`