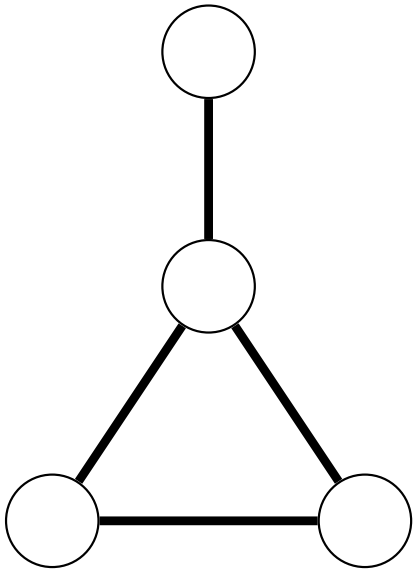
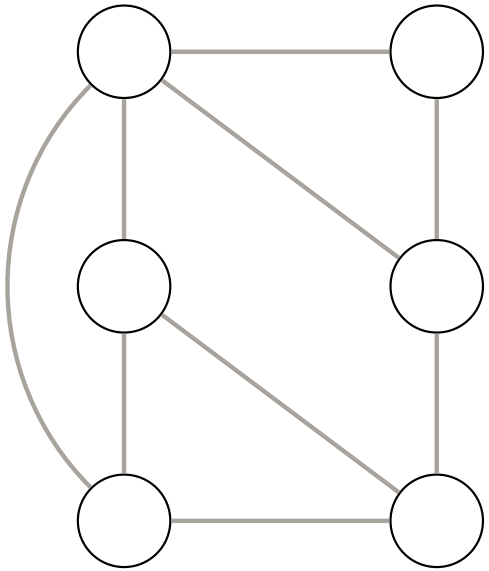
Are “Hard” Subgraph Problems Hard?

Ciaran McCreesh

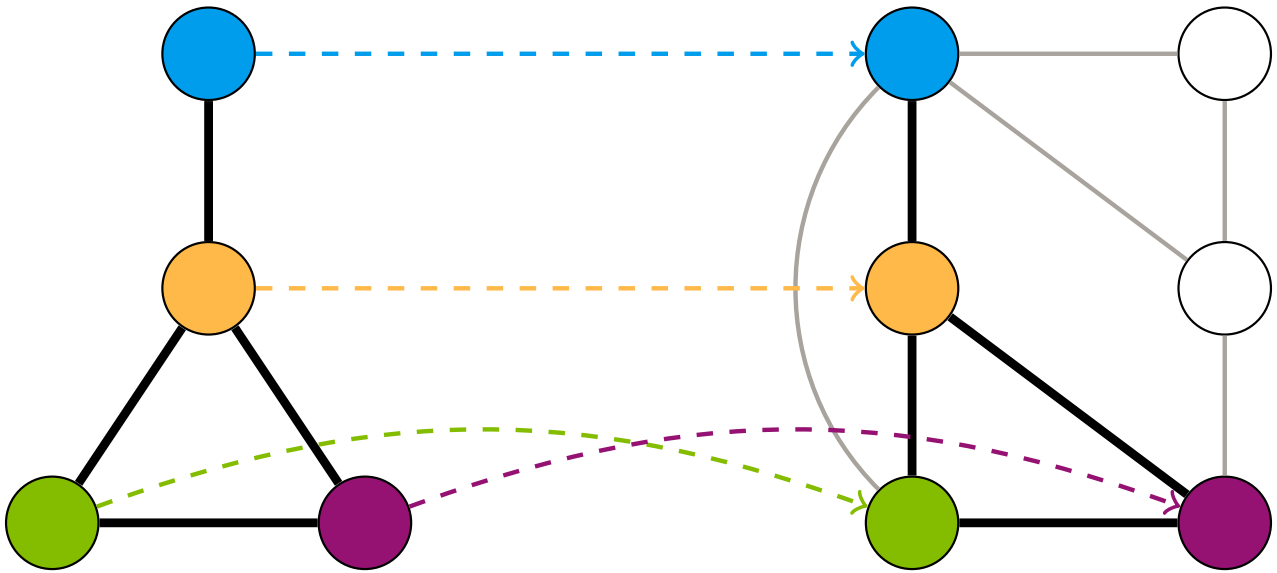
University of Glasgow

Subgraph Isomorphism

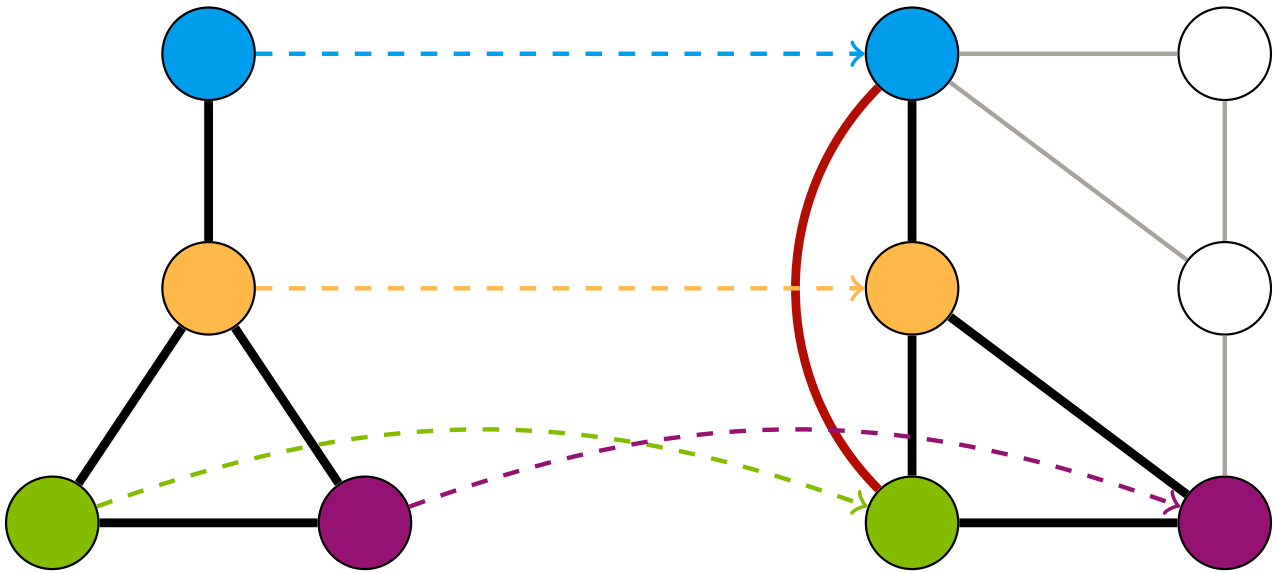




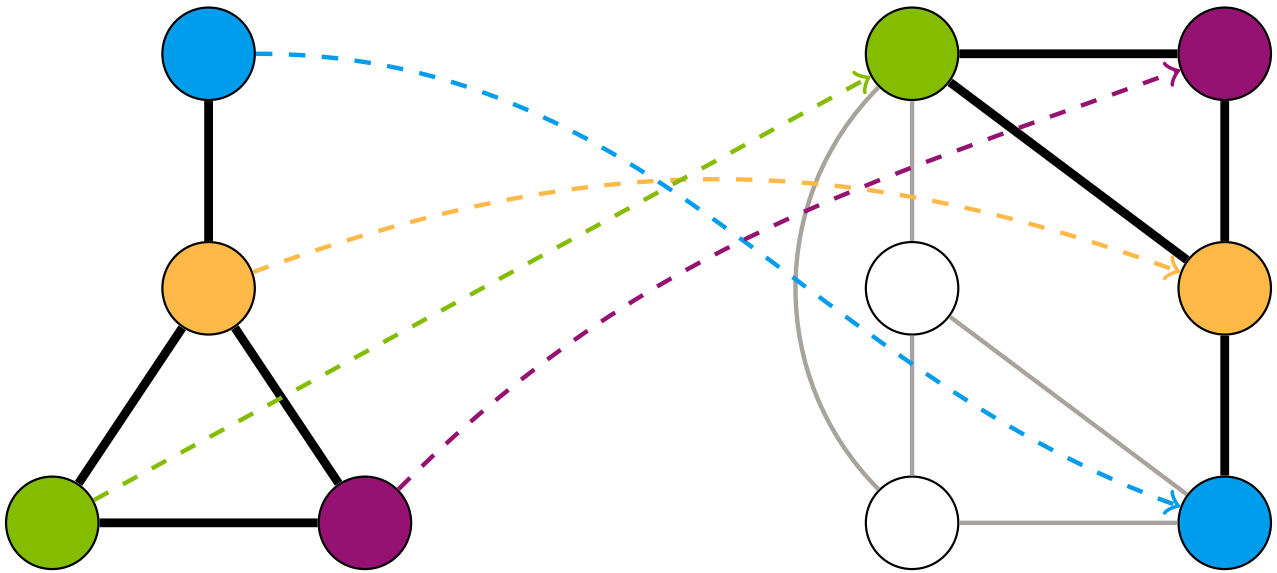
Subgraph Isomorphism



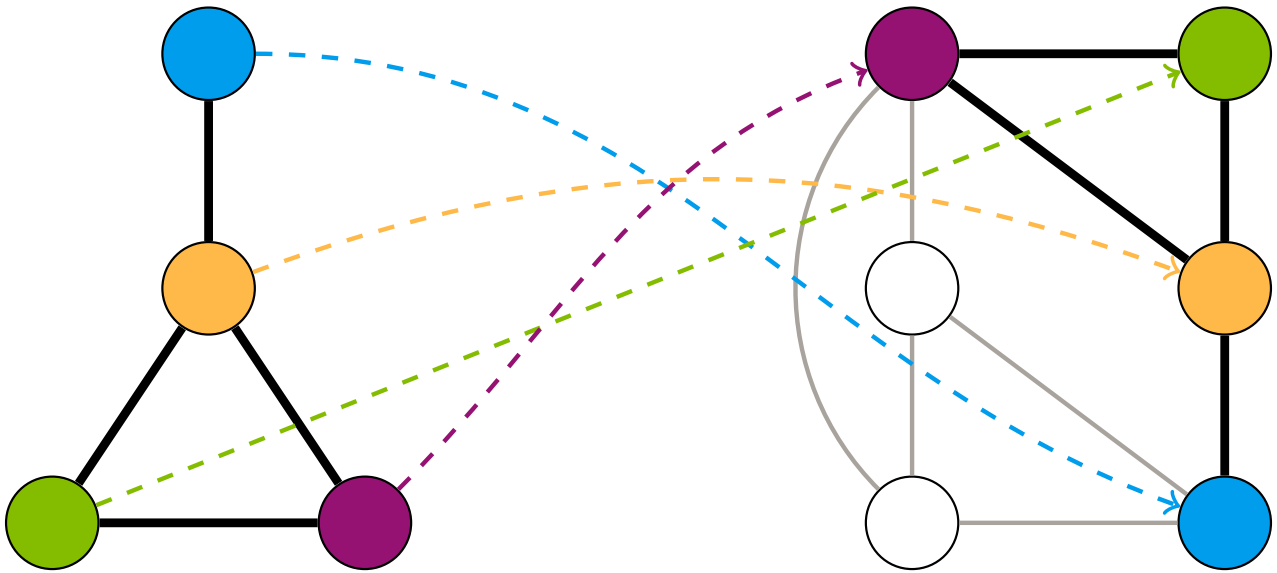
Subgraph Isomorphism



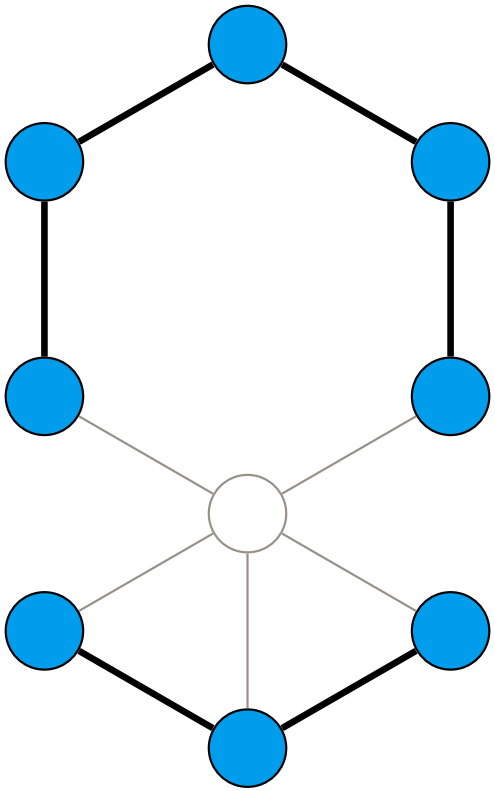
Subgraph Isomorphism

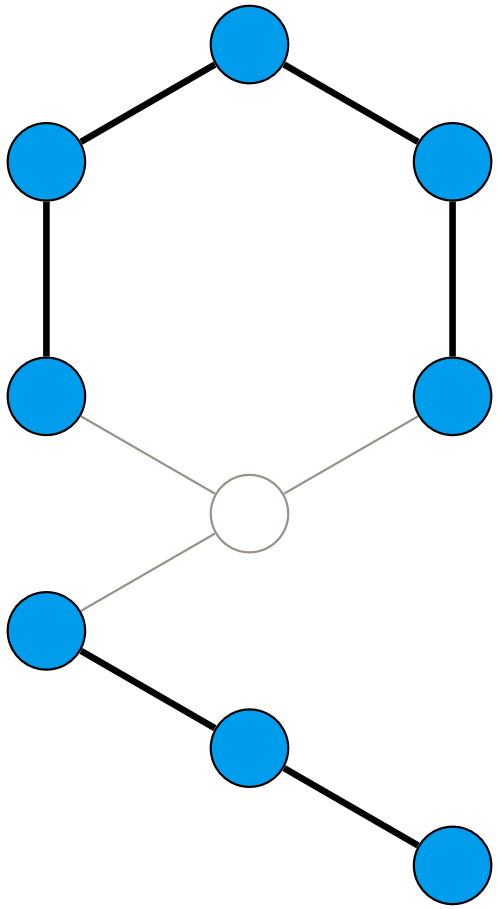


Subgraph Isomorphism

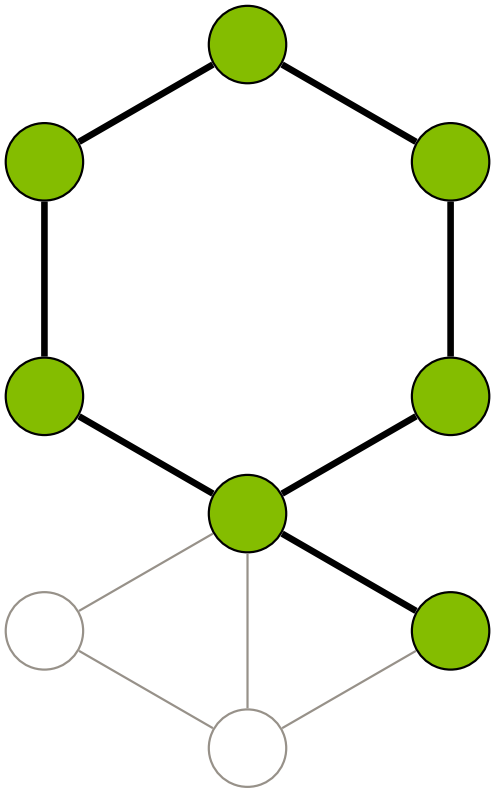


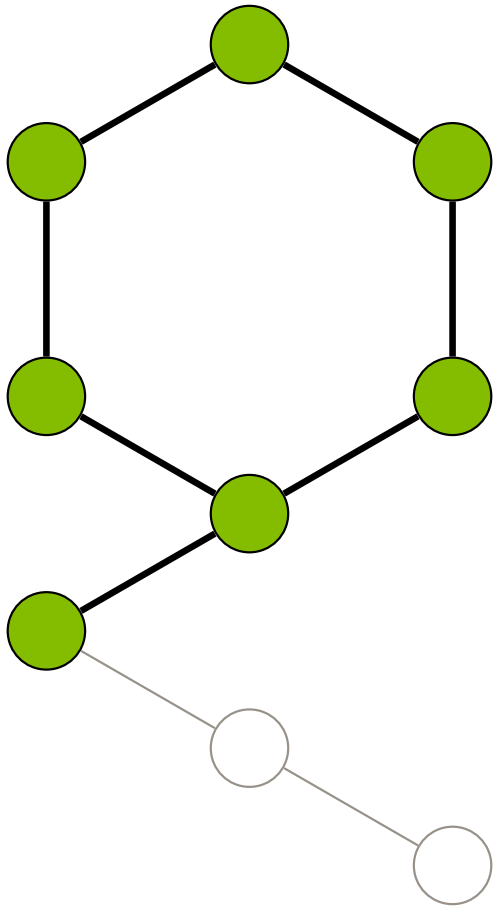
Maximum Common Induced Subgraph



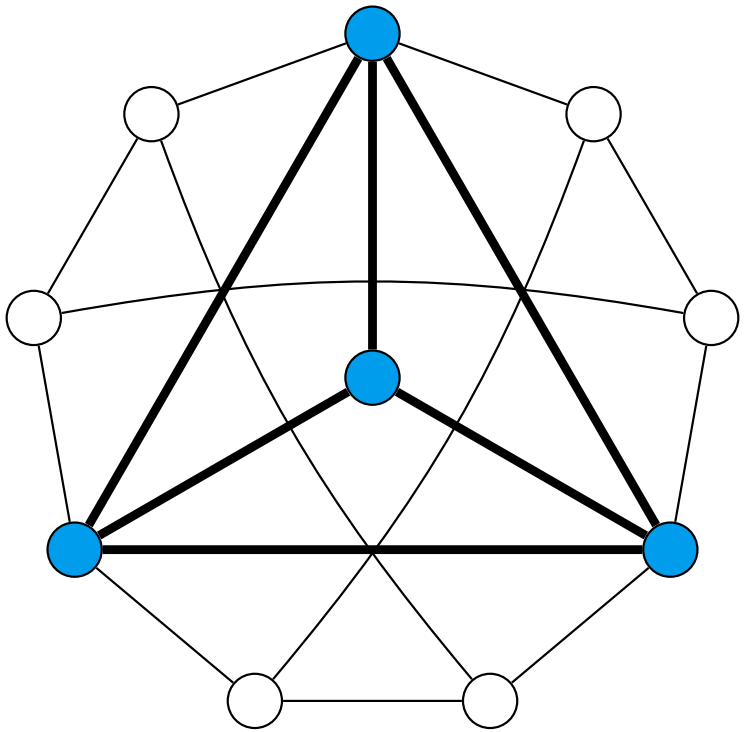


Maximum Common Induced Connected Subgraph





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.

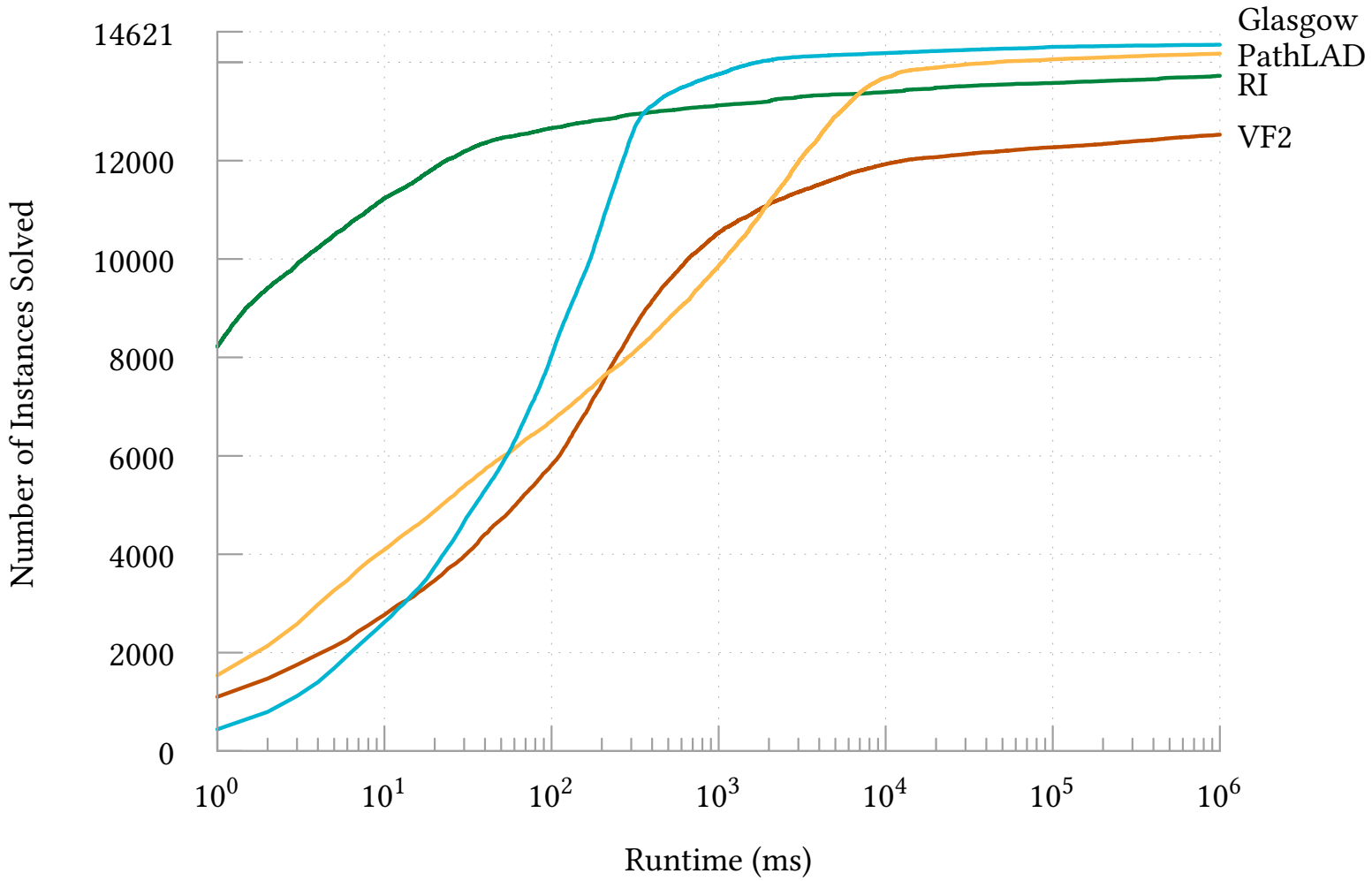
Popular Families of Subgraph Isomorphism Algorithms

- Connectivity-based:
  - VF2 (2004), VF3 (2017)
  - RI (2013)
- Constraint programming:
  - Ullman (1976)
  - LAD (AIJ 2010), SND (CP 2014), PathLAD (LION 2016)
  - Glasgow (CP 2015, LION 2016, CPAIOR 2019, . . .)

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.

Horse Race!


Glasgow
PathLAD
RI
VF2
Number of Instances Solved
14621
12000
10000
8000
6000
4000
2000
0
10^0
10^1
10^2
10^3
10^4
10^5
10^6
Runtime (ms)

Easy Conclusion!

- CP is best!

An Observation about Certain Datasets

- All of the randomly generated instances from the MIVIA suites are satisfiable.
- The target graphs are randomly generated, and patterns are made by selecting random connected subgraphs and permuting them.
- These instances are usually rather easy…
- Many papers use *only* these instances for benchmarking.

A Different Easy Conclusion!

- CP is slow! RI is best!

A Quick Introduction to Constraint Programming

- A declarative way of describing (hard) problems.
- A set of variables, each of which has a (finite) domain of values.
- A set of constraints (in any form we like, so arbitrary arity, non-linear, etc).
- Combining inference and clever backtracking search, give each variable a value from its domain, such that all constraints are respected.

Subgraph Finding, 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.
  - Injectivity, known as “all different”.

The Glasgow Subgraph Solver

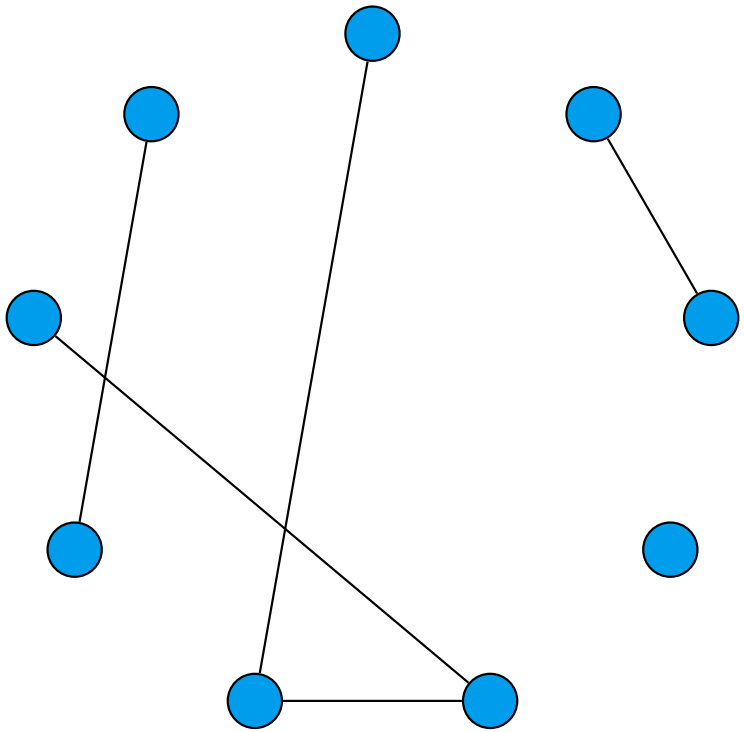
`https://github.com/ciaranm/glasgow-subgraph-solver`

- Subgraph isomorphism, and all its variants (induced / non-induced, homomorphism, locally injective, labels, side constraints, directed, ...).
- Also special algorithms for clique.

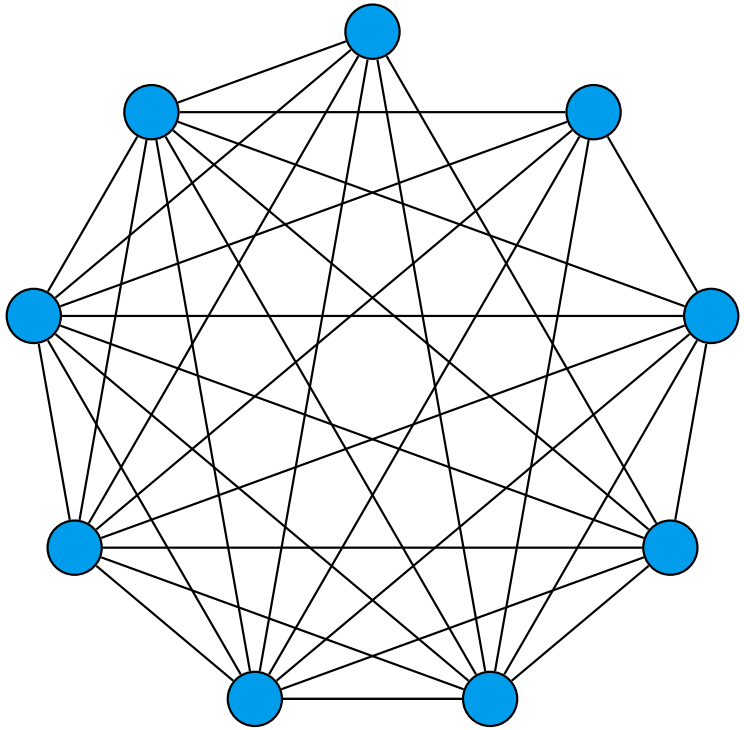
Some Clever Things We Can Deduce About Graphs

- A pattern vertex of degree k cannot be mapped to a target vertex of degree $k - 1$ or smaller.
  - We can also reason about neighbourhood degree sequences.
- Two pattern vertices that are distance d apart cannot be mapped to a pair of target vertices that are further than d apart.
  - We can also reason about the number of distinct short simple paths between two vertices.
- Search by branching on the smallest domain first, tie-breaking on the highest degree, and start by trying the highest degree target vertex first.
  - Then do more sneaky things involving slight randomisation, restarts, parallel search, ...
- If we have a clique subproblem, do something else.

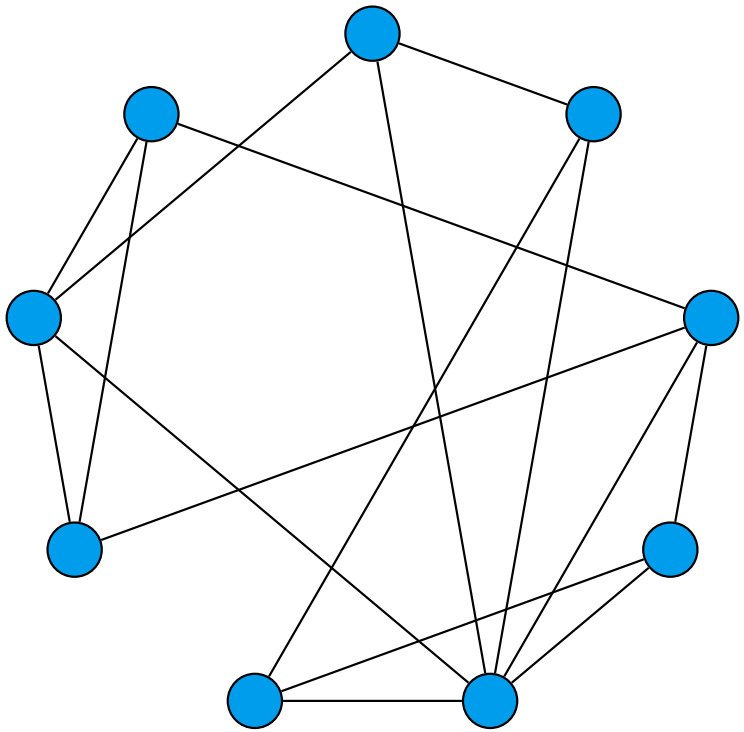
Is Clique-Finding Hard?



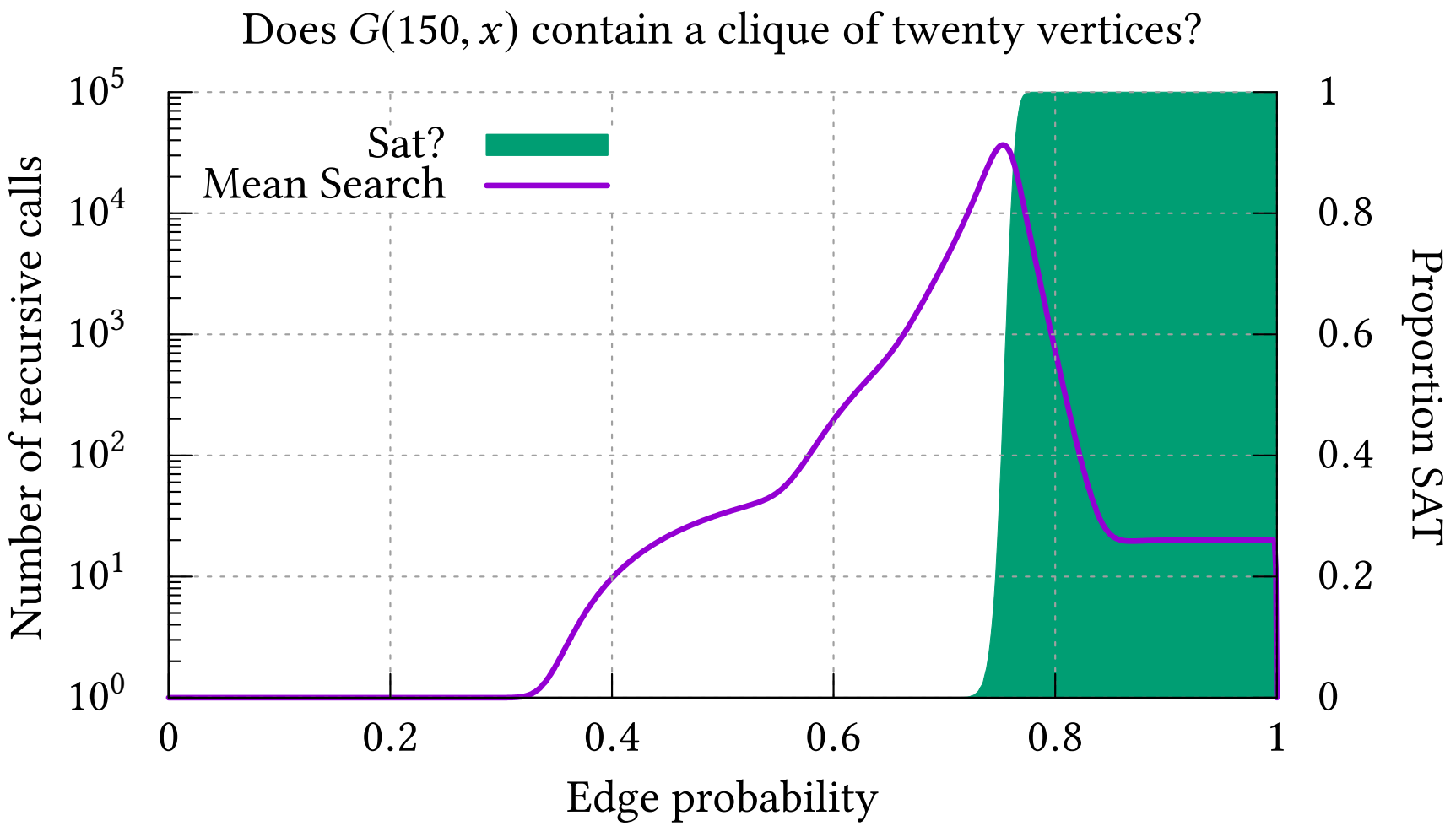
Is Clique-Finding Hard?



Is Clique-Finding Hard?



Cliques in Random Graphs

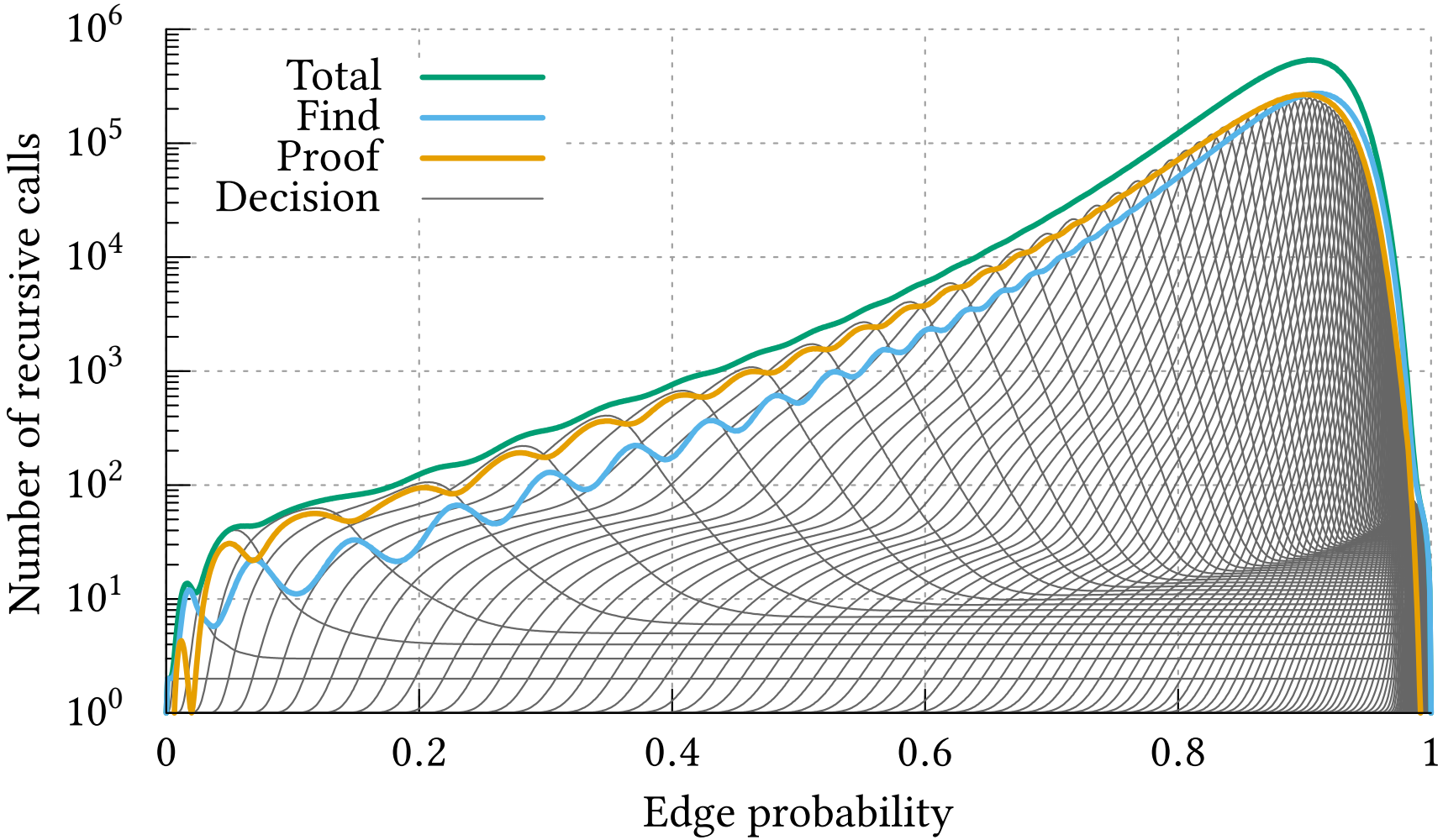


Intuition

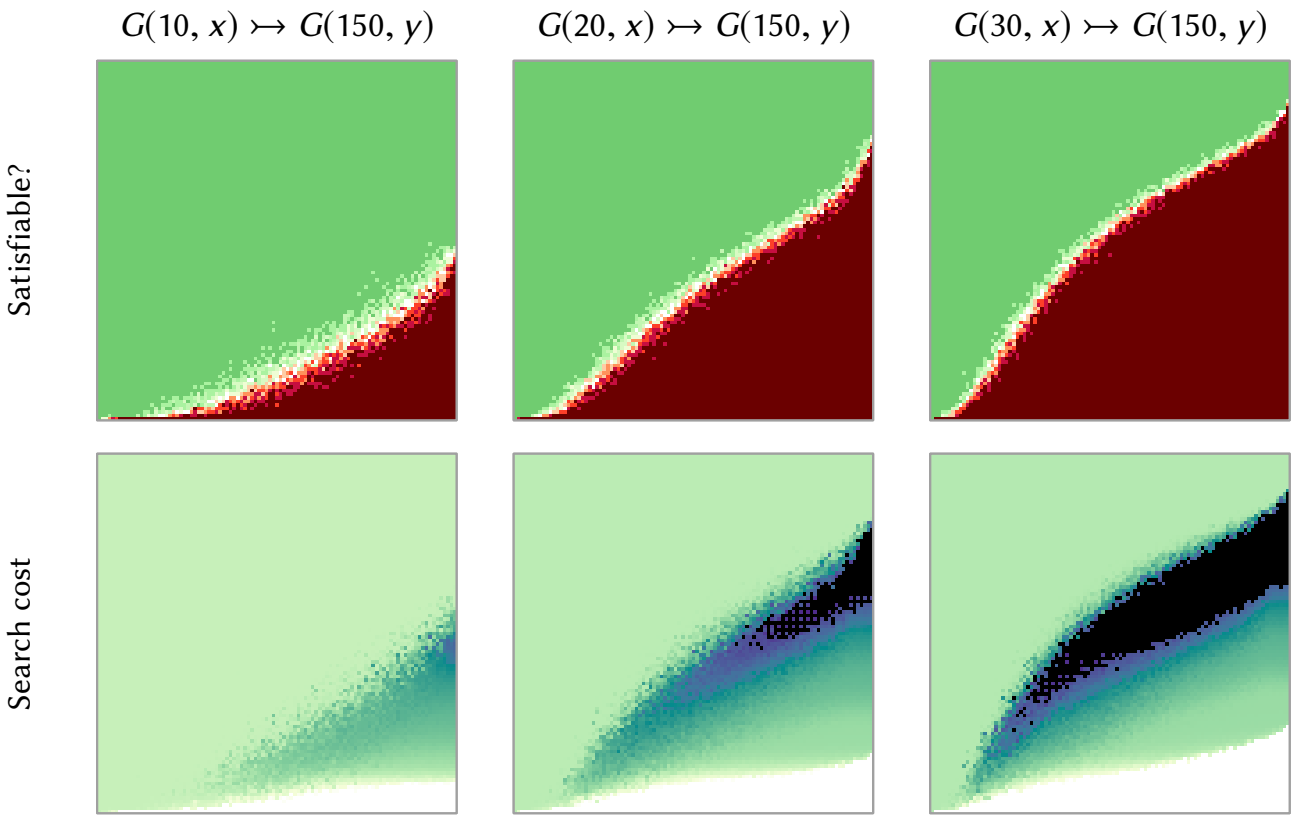
- High density means lots of occurrences, so wherever we look, it's easy to find one of them.[2]
- Low density means no occurrences, and we can quickly show we run out of edges after doing a bit of branching.
- If we expect there to be just one solution, it's really hard to find it if it exists, and really hard to rule it out if it doesn't exist.

[2] This statement is technically a massive lie.

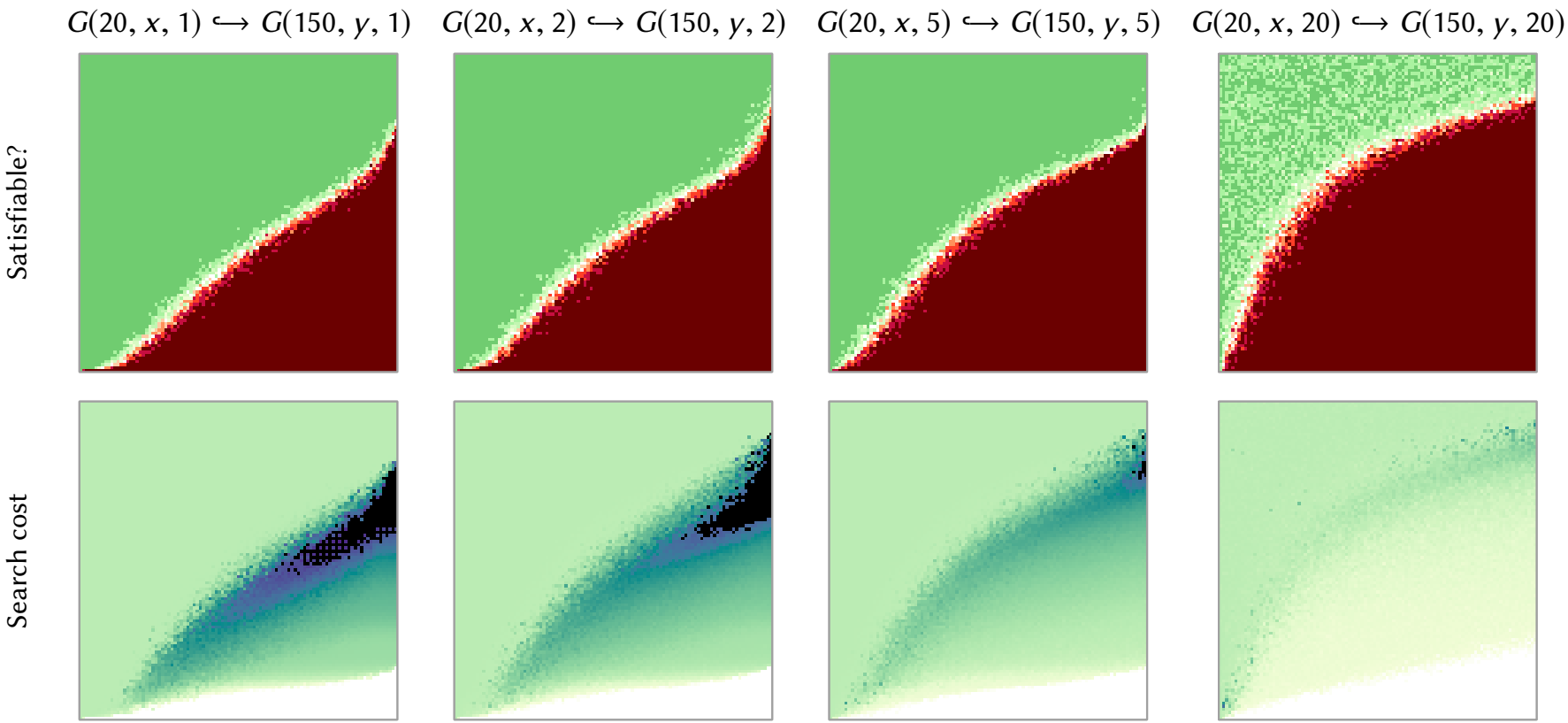
Optimisation?



Subgraph Isomorphism in Random Graphs



Labelled Graphs



Graph Databases

- Lots of labelled target graphs, more or less a fixed set.
- Patterns are presented in real time.
- Return every target graph that contains the pattern.

Filter / Verify

- A widely used technique in graph databases.
- NP-complete means subgraph isomorphism is slow. So why not avoid calling a subgraph isomorphism algorithm whenever possible?
- Detect "features" of graphs:
  - Number of times particular labels occur.
  - Whether or not certain small shapes occur.
  - Whether vertices of at least a particular degree exist.
- Only call a subgraph isomorphism algorithm on instances where the features match.

Filter / Verify

$$T = T_{search} + \left|C_q\right| \cdot T_{iso\_test}$$

"Graph indexing plays a critical role at efficient query processing in graph databases"[3]

[3] See McCreesh et al., JAIR 61(2018) for references

Filter / Verify

$$T_{search} + \left|C_q\right| \cdot (T_{io} + T_{iso\_test})$$

"the value of T_{iso_test} does not change much for a given query"[3]

[3] See McCreesh et al., JAIR 61(2018) for references

Filter / Verify

"Sequential scan is very costly because one has to not only access the whole graph database but also check subgraph isomorphism. It is known that subgraph isomorphism is an NP*-complete problem. Clearly, it is necessary to build graph indices in order to help processing graph queries."*[3]

[3] See McCreesh et al., JAIR 61(2018) for references

Filter / Verify

"obviously it is inefficient to perform a sequential scan on every graph in the database, because the subgraph isomorphism test is expensive"[3]

[3] See McCreesh et al., JAIR 61(2018) for references

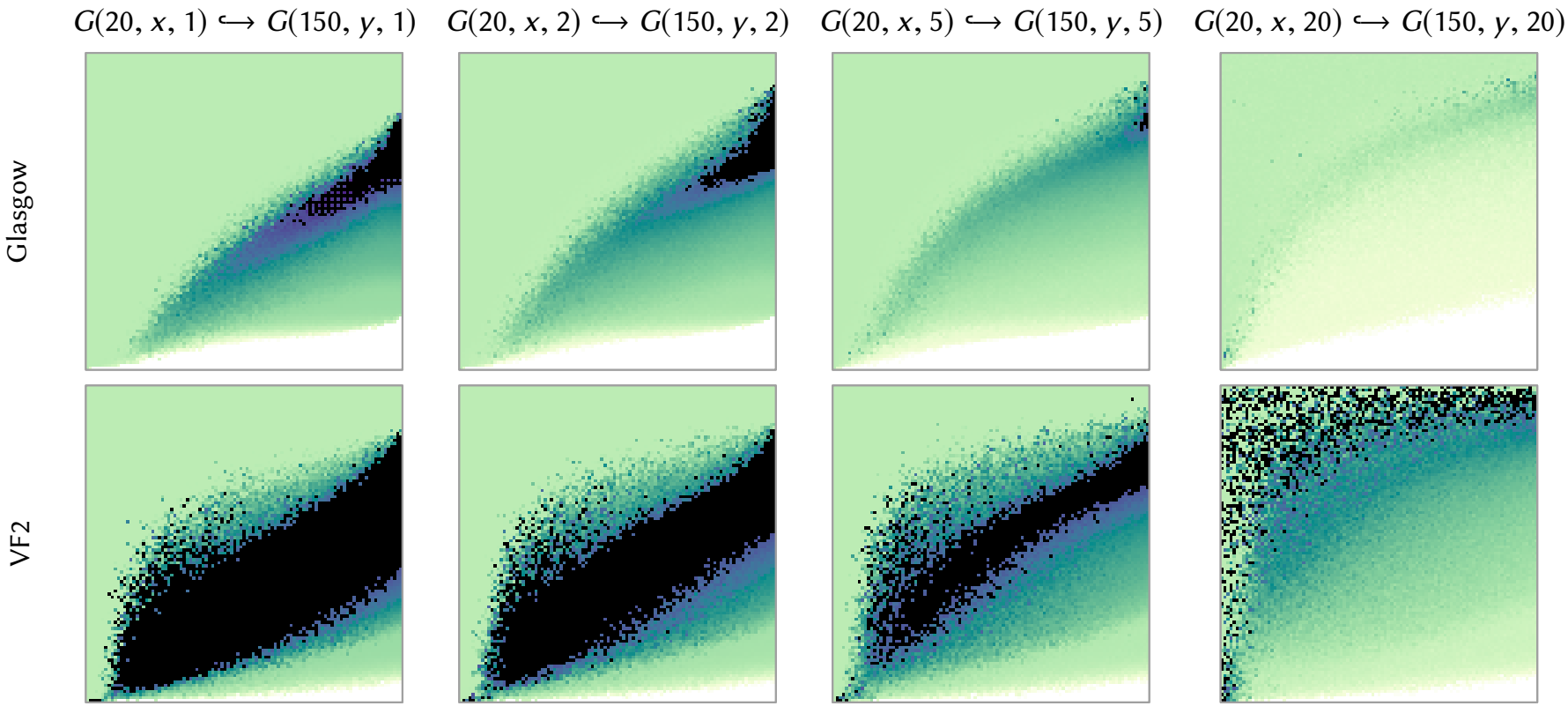
Filter / Verify

"the graph query problem is hard in that . . . it requires subgraph isomorphism checking . . . which has proven to be NP*-complete"*

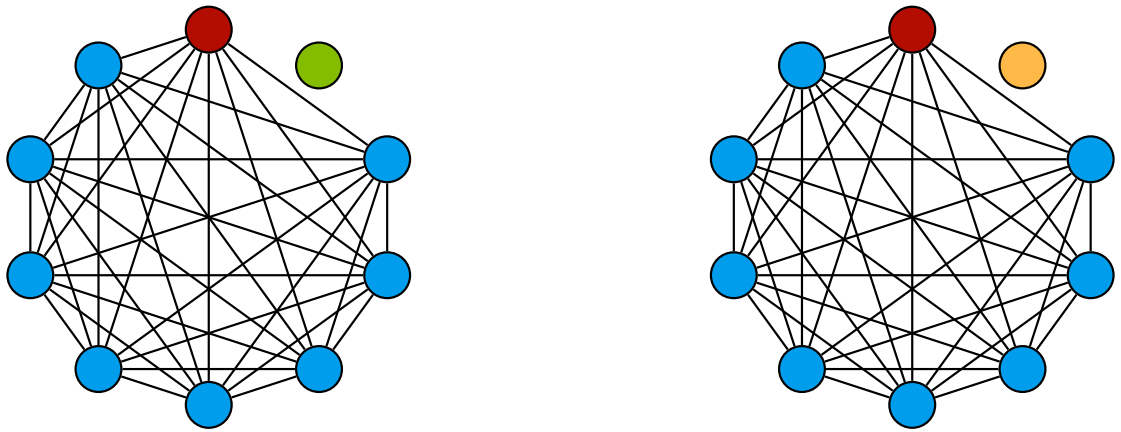
working with large networks is hard or impossible "due to the lack of scalable graph indexing mechanisms"[3]

[3] See McCreesh et al., JAIR 61(2018) for references

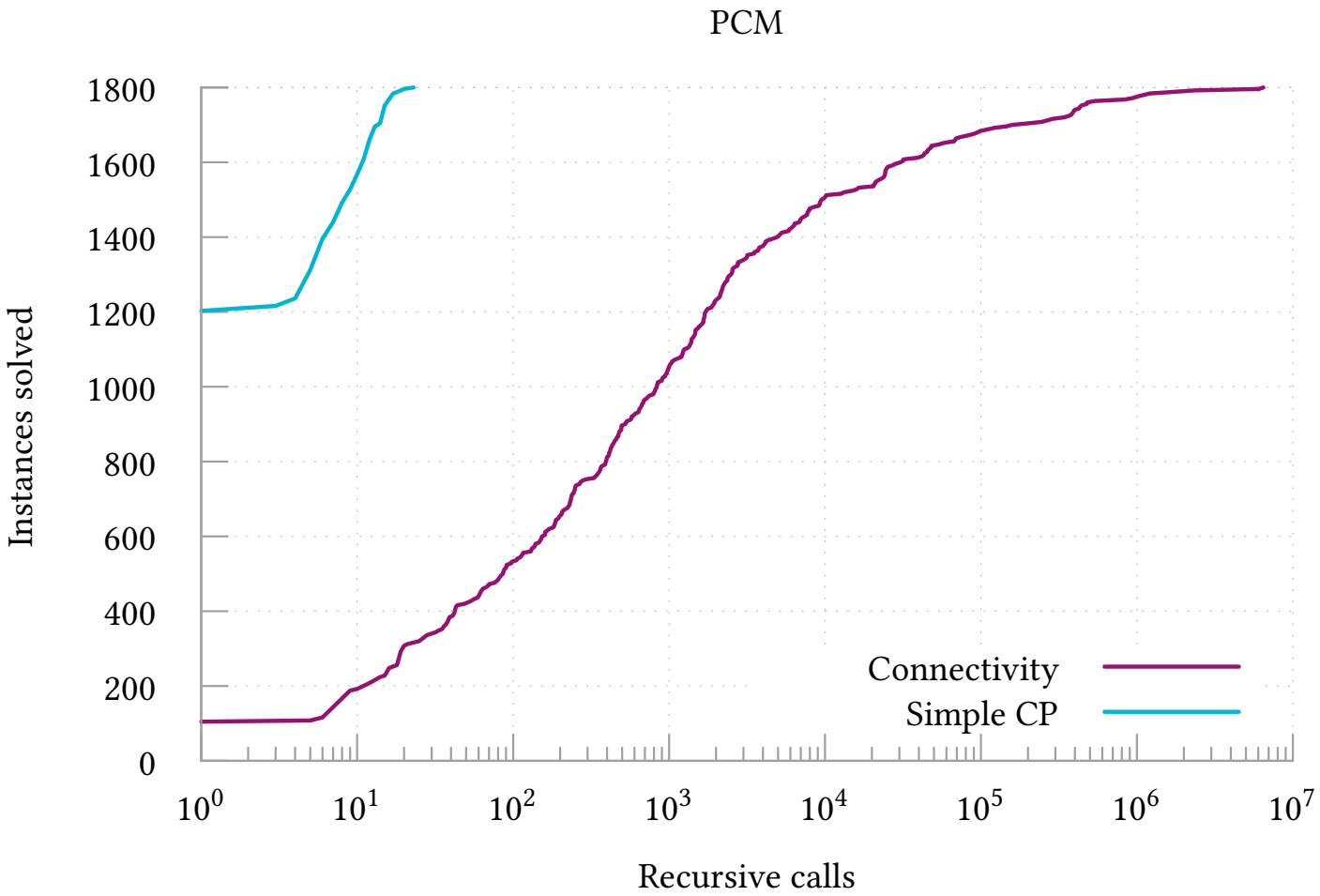
Connectivity Algorithms are Really Stupid



Connectivity Algorithms are Really Stupid



Constraint Programming versus Connectivity



It's Not Just Graph Databases

"There are (not so many) instances for which creation of a clear design is prohibitively slow in the current implementation that evaluates subgraph isomorphism with the VF2 algorithm ... Recent experiences with a few of these showed that the LAD algorithm was very fast in ruling out impossible matches, where VF2 took a long time."[4]

[4] See McCreesh et al., JAIR 61(2018) for references

So What?

- There's a lot more to hardness than worst-case complexity analysis.
- Understanding how algorithms behave is important.
- Maybe we should try doing some science?

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

University of Glasgow