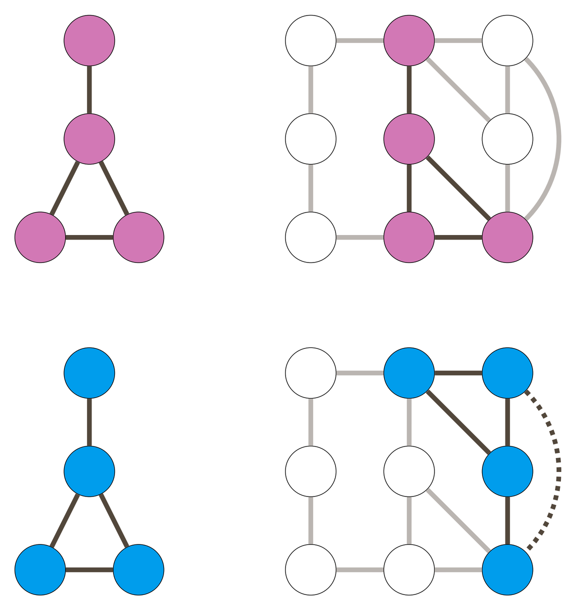
Heuristics and Really Hard Instances for Subgraph Isomorphism Problems

Ciaran McCreesh, Patrick Prosser and James Trimble, University of Glasgow, Glasgow, Scotland

We show how to generate **really hard** random instances for **subgraph isomorphism** problems. For the non-induced variant, we predict and observe a phase transition between satisfiable and unsatisfiable instances, with a corresponding complexity peak seen in **three different solvers**. For the induced variant, much richer behaviour is observed, and **constrainedness** gives a better measure of difficulty than does proximity to a phase transition. We also discuss **variable and value ordering** heuristics, and their relationship to the **expected number of solutions**.

Subgraph Isomorphism (Finding Patterns in Graphs)

The **non-induced subgraph isomorphism problem** is to find an injective mapping from a given pattern graph to a given target graph which preserves adjacency—in essence, we are "finding a copy of" the pattern inside the target.



The **induced** variant of the problem additionally requires that the mapping preserve non-adjacency, so there are no "extra edges" in the copy of the pattern that we find. The top example is induced, whereas the bottom example is not, due to the dashed edge.

Despite these problems being NP-complete, modern practical subgraph isomorphism algorithms can handle problem instances with many hundreds of vertices in the pattern graph, and up to **ten thousand vertices** in the target graph, leading to successful application in areas such as **computer vision**, **biochemistry**, and **pattern recognition**. However, these algorithms cannot handle arbitrary instances of this size—here we generate instances with thirty and one hundred and fifty vertices respectively which cannot be solved within one hour.

The Expected Number of Solutions, and Heuristics

If we randomly generate a pattern graph with p vertices and density d_p, and a target graph with t vertices and density d_t, the **expected number of solutions** to the non-induced isomorphism problem is

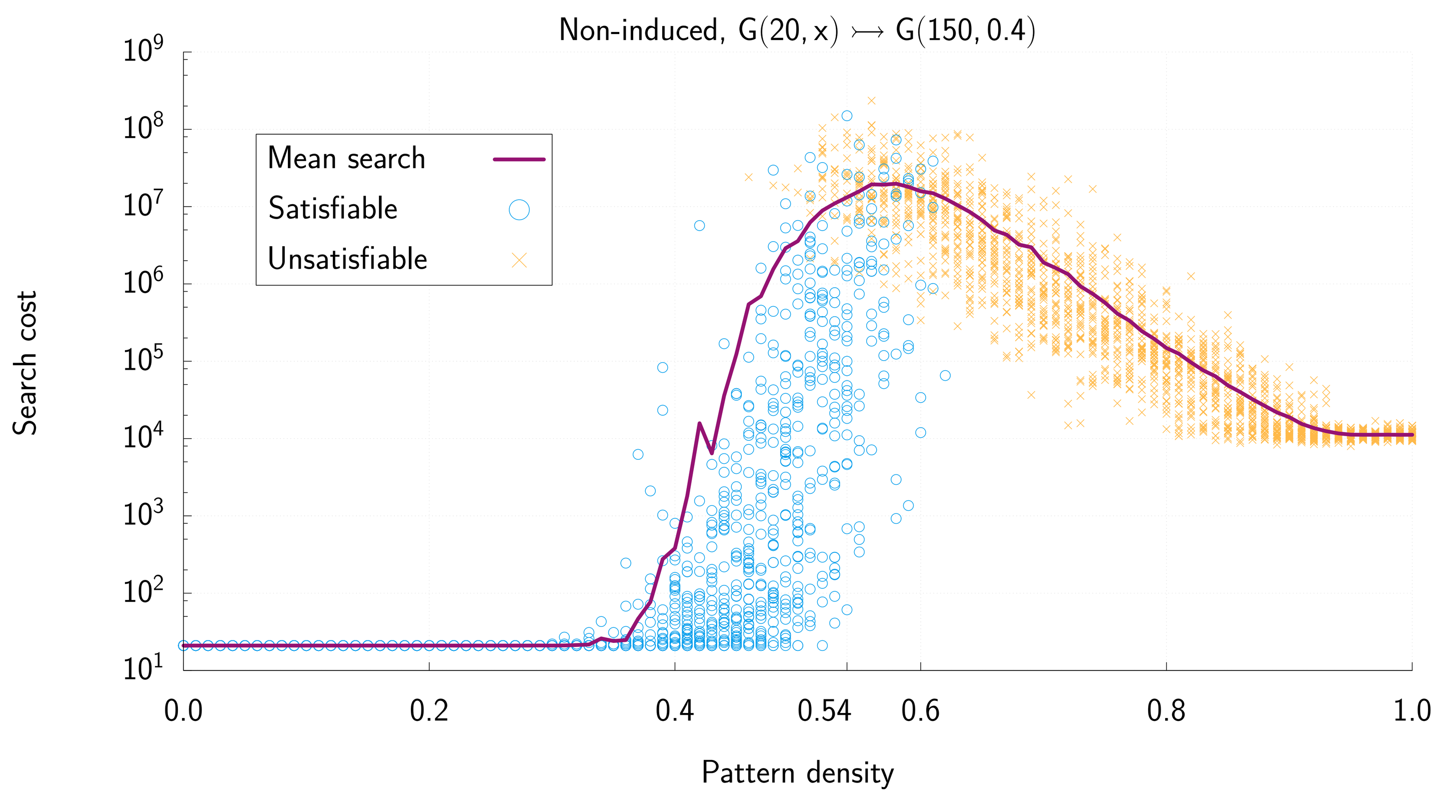
$$\langle \mathrm{Sol} \rangle = t \cdot (t-1) \cdot \ldots \cdot (t-p+1) \cdot d_t^{\,d_p \cdot \binom{p}{2}}.$$

By considering when $\langle \mathrm{Sol} \rangle = 1$, we can **estimate the location** of the satisfiable / unsatisfiable phase transition. (This is not entirely accurate due to variance, particularly when the pattern is very sparse or very dense).

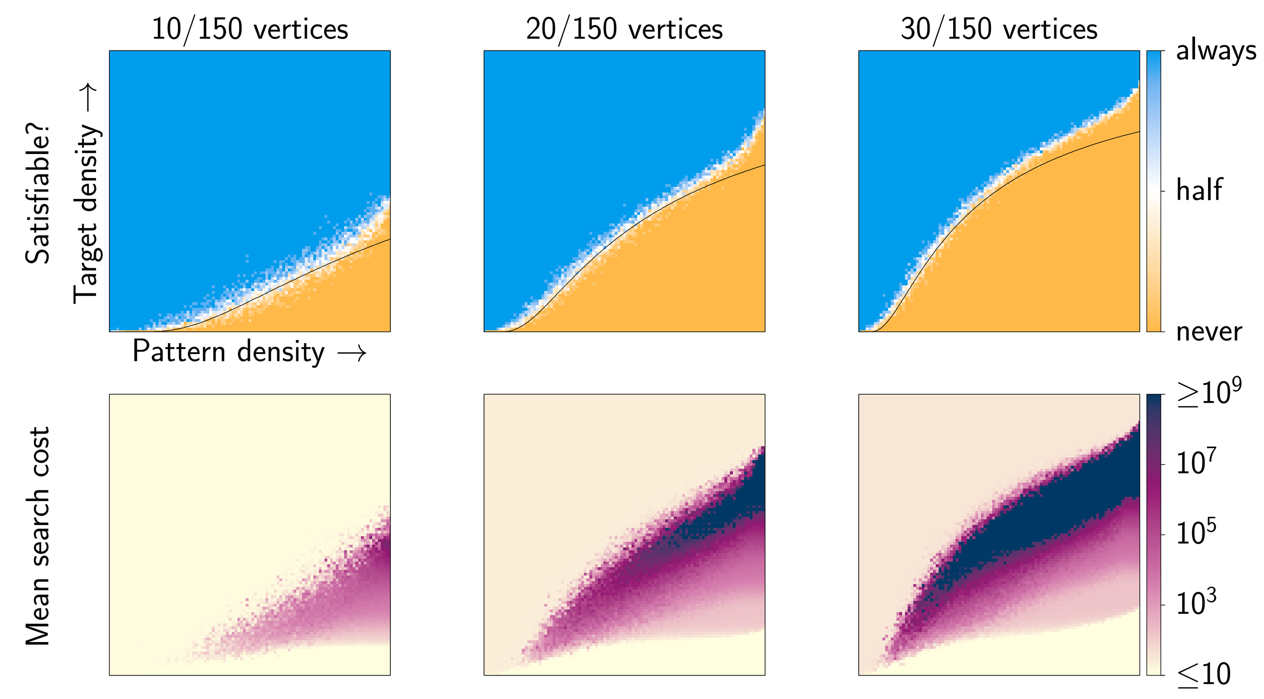
We can also use this formula to recover **variable** and **value ordering heuristics**: we should make branching decisions which maximise the expected number of solutions during search. This tells us to prefer small domains and pattern vertices of high degree, but target vertices of low degree, which matches the empirically-determined heuristics used in state of the art solvers.

Non-Induced Phase Transitions

If we generate random patterns and targets independently, fixing a pattern order of 20 vertices, a target order of 150 vertices, a target density of 0.4, and varying the pattern density, the familiar phase transition and (solver-independent) complexity peak occurs.



We can also look at what happens if we vary both the pattern and the target density, for different orders of pattern. The top row shows satisfiability—the black lines show where $\langle \mathrm{Sol} \rangle = 1$. The bottom row shows the search cost for one of the three algorithms we consider in the paper. As expected, the "really hard" instances are near the phase transition.



Induced Phase Transitions

For induced isomorphisms, the behaviour is much richer. We can derive an expected number of solutions of

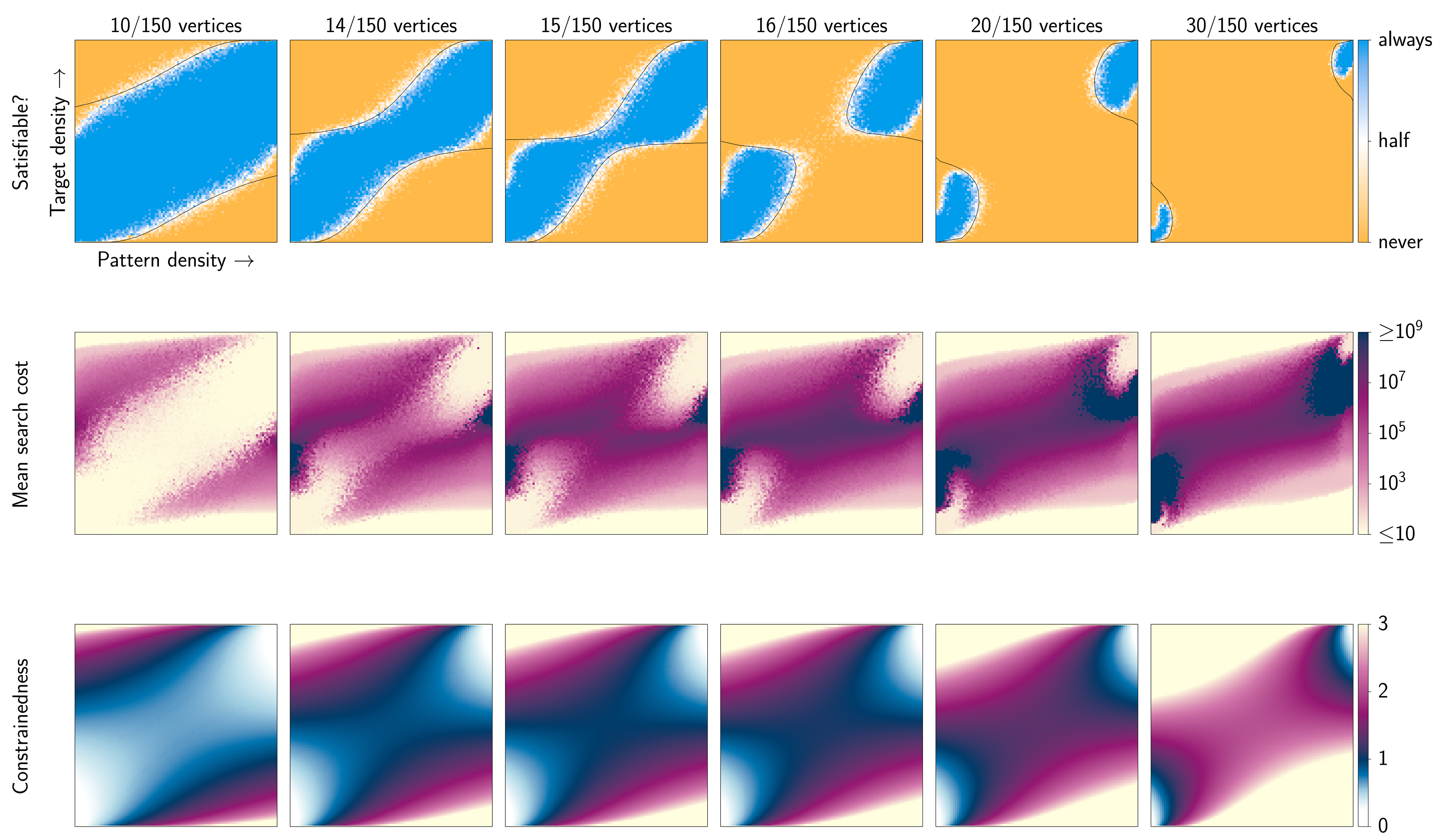
$$\langle \mathrm{Sol} \rangle = t \cdot (t-1) \cdot \ldots \cdot (t-p+1) \cdot d_t^{\,d_p \cdot \binom{p}{2}} \cdot (1-d_t)^{(1-d_p)\cdot \binom{p}{2}}.$$

Again this predicts a sharp phase transition. When the pattern is small, we see hard instances near the phase transition, and easy instances elsewhere. However, when the pattern is larger, the central unsatisfiable region **remains very hard**, even though it is no longer near a phase transition.

This is not just a weakness of current subgraph isomorphism algorithms: the region is also hard for pseudo-boolean and boolean satisfiability solvers, and under reduction to the clique problem.

The central hard region *is* predicted correctly by **constrainedness**, as we show in the third row of plots. Although far from a phase transition, this region is only slightly overconstrained.

Looking to maximise $\langle \mathrm{Sol} \rangle$ derives existing variable and value ordering heuristics, in the case that both the pattern and target graphs are sparse. The formula suggests that algorithms should swap heuristics when the pattern and target are dense—empirically, this does indeed give an improvement. However, maximising the expected number of solutions is **not enough to select the best heuristic** in two eighths of the search space (and constrainedness does not help).



See the Paper For. . .

- Results from two other subgraph isomorphism algorithms.
- Behaviour under reduction to clique, pseudo-boolean, and boolean satisfiability.
- Should we alter heuristics based upon the expected number of solutions?

Future Work

- Variance for more accurate predictions (if you can help, please get in touch).
- Other random models (degree-bounded, scale-free, . . .).
- Labelled graphs.

This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/K503058/1].

c.mccreesh.1@research.gla.ac.uk