# Heuristics and Really Hard Instances for Subgraph Isomorphism Problems

**Ciaran McCreesh*** and **Patrick Prosser** and **James Trimble**
University of Glasgow, Glasgow, Scotland
c.mccreesh.1@research.gla.ac.uk

## Abstract

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.

## 1 Introduction

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. We illustrate both variants in Figure 1. 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 [Cordella *et al.*, 2004; Solnon, 2010; Audemard *et al.*, 2014; McCreesh and Prosser, 2015], leading to successful application in areas such as computer vision [Damiand *et al.*, 2011; Solnon *et al.*, 2015], biochemistry [Giugno *et al.*, 2013; Carletti *et al.*, 2015], and pattern recognition [Conte *et al.*, 2004].

However, these algorithms cannot handle *arbitrary* instances of this size. The experimental evaluations of these algorithms were performed using a mix of real-world graphs, graphs that encode biochemistry and computer vision problems, and randomly generated graph pairs. Using random instances to evaluate algorithm behaviour can be beneficial, because it provides a way of generating many instances cheaply, and reduces the risk of over-fitting when tuning design parameters. The random instances used in each case came from common datasets [De Santo *et al.*, 2003; Zampelli *et al.*, 2010], which were generated by taking a random subgraph of a random (Erdős-Rényi, scale-free, bounded degree, or mesh) graph and permuting the vertices. Such instances are guaranteed to be satisfiable—Anton and Olson [2009] exploited this property to create large sets of random satisfiable boolean satisfiability instances. However, since this has been the only approach used to generate random subgraph isomorphism instances, existing benchmark suites contain relatively few non-trivial unsatisfiable instances, and the satisfiable instances tend to be computationally fairly easy, with most of the difficulty being in dealing with the size of the model. This has lead to bias in algorithm design, to the extent

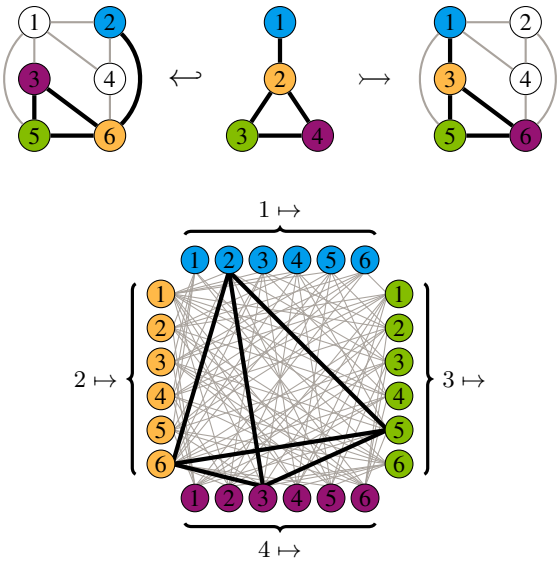


Figure 1: On top, there is a non-induced isomorphism from the pattern graph, in the center, to the target on the right, mapping vertex 1 to 1, 2 to 3, 3 to 5 and 4 to 6. This is not an induced isomorphism, since there is an edge between 1 and 5 in the target but not between 1 and 3 in the pattern. The mapping from the pattern to the (same) target on the left, sending 1 to 2, 2 to 6, 3 to 5 and 4 to 3, is both non-induced and induced. Below, the association graph encoding for the induced version: the highlighted clique corresponds to the same mapping.

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

that some proposed techniques will *only* work on satisfiable instances [Battiti and Mascia, 2007].

The lack of unsatisfiable instances cannot be addressed by using a pattern graph from one of the random suites with the "wrong" target graph: this tends to give either a trivially unsatisfiable instance, or a satisfiable instance. (In particular, it is *not* the case that a relatively small random graph is unlikely to appear in a larger random graph.)

Here we present and evaluate a new method for creating random pattern/target pairs. This method generates both satisfiable and unsatisfiable instances, and can produce computationally challenging instances with only a few tens of vertices in the pattern, and 150 vertices in the target. Our work builds upon the phase transition phenomena observed in satisfiability and graph colouring problems first described by Cheeseman *et al.* [1991] and Mitchell *et al.* [1992]. For subgraph isomorphism we identify three relevant control parameters: we can independently alter the edge probability of the pattern graph, the edge probability of the target graph, and the relative orders (number of vertices) of the pattern and target graphs. For non-induced isomorphisms, with the correct choice of parameters we see results very similar to those observed with boolean satisfiability problems: there is a phase transition (whose location we can predict) from satisfiable to unsatisfiable, we see a solver-independent complexity peak occur near this phase transition, and understanding this behaviour helps us to select variable and value ordering heuristics.

For certain choices of parameters for induced isomorphisms, there are two phase transitions, going from satisfiable to unsatisfiable, and then from unsatisfiable back to satisfiable. Again, when going from satisfiable to unsatisfiable (from either direction), instances go from being trivial to really hard to solve. However, each of the three solvers we tested also finds the central unsatisfiable region to be hard, despite it not being near a phase transition. To show that this is not a simple weakness of current subgraph isomorphism algorithms, we verify that this region is also hard when using a pseudo-boolean encoding, and under reduction to the clique problem. Interestingly, the constrainedness measure proposed by Gent *et al.* [1996b] *does* predict this difficult region—these instances provide evidence in favour of constrainedness, rather than proximity to a phase transition, being an accurate predictor of difficulty, and show that constrainedness is not simply a refinement of a phase transition prediction.

### 1.1 Definitions

Throughout, our graphs are unlabelled, undirected, and do not have any loops. The *order* of a graph is the cardinality of its vertex set. We write $\mathrm{V}(G)$ for the vertex set of a graph $G$. The *complement* of a graph $G$, denoted $\overline{G}$, is the graph with the same vertex set as $G$, and with an edge between distinct vertices $v$ and $w$ if and only if $v$ and $w$ are not adjacent in $G$. We write $G(n, p)$ for an Erdős-Rényi random graph with $n$ vertices, and an edge between each distinct pair of vertices with independent probability $p$.

A *non-induced subgraph isomorphism* from a graph $P$ (called the *pattern*) to a graph $T$ (called the *target*) is an injective mapping from $\mathrm{V}(P)$ to $\mathrm{V}(T)$ which preserves adjacency—that is, for every adjacent $v$ and $w$ in $\mathrm{V}(P)$, the vertices $i(v)$ and $i(w)$ are adjacent in $T$. An *induced subgraph isomorphism* additionally preserves non-adjacency—that is, if $v$ and $w$ are not adjacent in $P$, then $i(v)$ and $i(w)$ must not be adjacent in $T$. We use the notation $i : P \rightarrowtail T$ for a non-induced isomorphism, and $i : P \hookrightarrow T$ for an induced isomorphism. Observe that an induced isomorphism $i : P \hookrightarrow T$ is a non-induced isomorphism $i : P \rightarrowtail T$ which is also a non-induced isomorphism $i : \overline{P} \rightarrowtail \overline{T}$.

### 1.2 Experimental Setup

Our experiments were performed on systems with Intel Xeon E5-4650 v2 CPUs, running Scientific Linux release 6.7. We selected three subgraph isomorphism solvers: the Glasgow solver [McCreesh and Prosser, 2015], LAD [Solnon, 2010], and VF2 [Cordella *et al.*, 2004]; each was compiled using GCC 4.9.

The Glasgow and LAD solvers use backtracking search to build up an assignment of target vertices (values) to pattern vertices (variables), but differ in terms of inference and ordering heuristics. The approach used by VF2 is similar, although the domains of variables are not stored (in the style of conventional backtracking, rather than forward-checking), and so domain wipeouts are not detected until an assignment is made.

In each case we measure the number of recursive calls (guessed assignments) made, not runtimes. We are not aiming to compare absolute performance between solvers; rather, we are looking for solver-independent patterns of difficulty. We used a timeout of 1,000 seconds, which was enough for the Glasgow solver to solve nearly all our instances (whose orders were selected with this timeout in mind), although we may slightly overestimate the proportion of unsatisfiable instances for extremely sparse or dense pattern graphs. The LAD and VF2 solvers experienced many more failures with this timeout, so our picture of just how hard the hardest instances are with these solvers is less detailed.

## 2 Non-Induced Subgraph Isomorphisms

Suppose we arbitrarily decide upon a pattern graph order of 20, a target graph order of 150, and a fixed target edge probability of 0.40. As we vary the pattern edge probability from 0 to 1, we would expect to see a shift from entirely satisfiable instances (with no edges in the pattern, we can always find a match) to entirely unsatisfiable instances (a maximum clique in this order and edge probability of target graph will usually have between 9 and 12 vertices). The dark line in Figure 2 shows that this is the case. For densities of 0.67 or greater, no instance is satisfiable; with densities of 0.44 or less, every instance is satisfiable; and with a density of 0.55, roughly half the instances are satisfiable.

The light line plots mean search effort using the Glasgow solver: for sparse patterns, the problem is trivial, for dense patterns proving unsatisfiability is not particularly difficult, and we see a complexity peak around the point where half the instances are satisfiable. We also plot the search cost of individual instances, as points. The behaviour we observe looks remarkably similar to random 3SAT problems—compare, for example, Figure 1 of Leyton-Brown *et al.* [2014]. In particular, satisfiable instances tend to be easier, but show greater

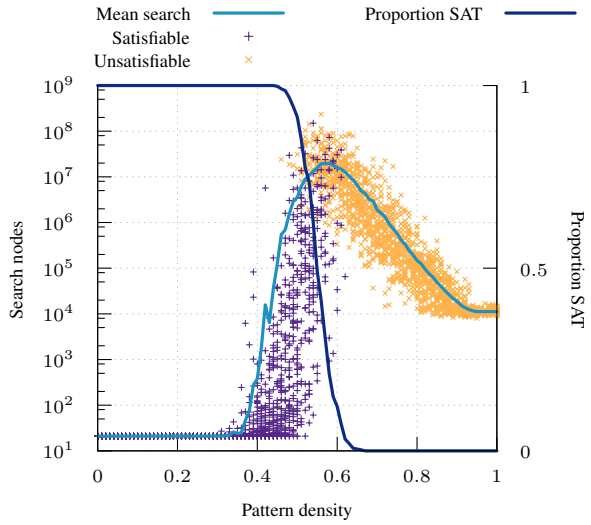


Figure 2: With a fixed pattern graph order of 20, a target graph order of 150, a target edge probability of 0.40, and varying pattern edge probability, we observe a phase transition and complexity peak with the Glasgow solver in the non-induced variant. Each point represents one instance. The lines show mean search effort and mean proportion satisfiable.

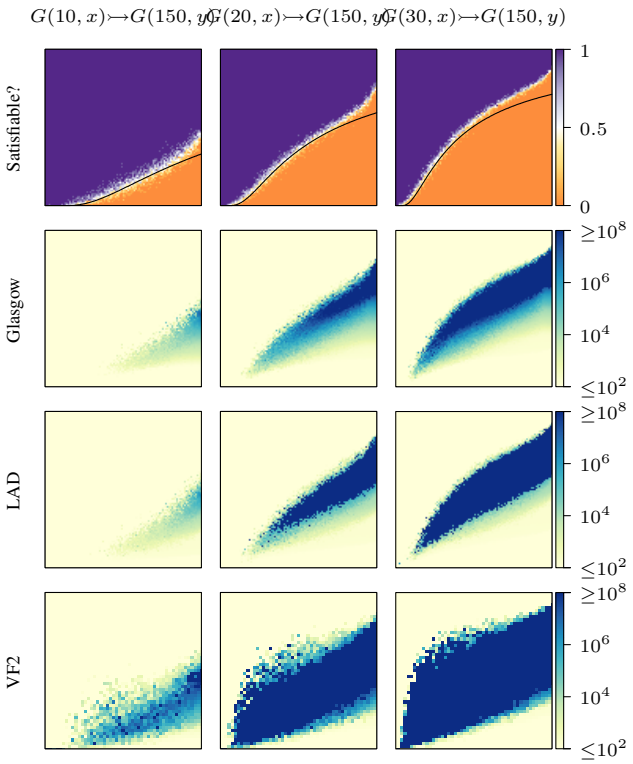


Figure 3: Behaviour of algorithms on the non-induced variant. For each plot, the x-axis is the pattern edge probability and the y-axis is the target edge probability, both from 0 to 1. Along the top row, we show the proportion of instances which are satisfiable; the white bands shows the phase transitions, and the black lines are our predictions of where the phase transition will occur. On the final three rows, we show the number of search nodes used by the Glasgow, LAD and VF2 solvers; the dark regions indicate "really hard" instances.

variation than unsatisfiable instances, and there are exceptionally hard satisfiable instances [Smith and Grant, 1997]. (The Glasgow solver supports parallel search with a work-stealing strategy explicitly designed to eliminate these. We have not enabled this option to avoid dealing with the complexity of search tree measurements under parallelism.)

What if we alter the edge probabilities for both the pattern graph and the target graph? In the top row of Figure 3 we show the satisfiability phase transition for the non-induced variant, for patterns of order 10, 20 and 30, targets of order 150, and varying pattern (x-axis) and target (y-axis) edge probabilities. Each axis runs over 101 edge probabilities, from 0 to 1 in steps of 0.01, except for the VF2 row which uses steps of 0.02. For each of these points, we generate ten random instances. The colour denotes the proportion of these instances which were found to be satisfiable. Inside the orange region, at the bottom right of each plot, every instance is unsatisfiable—here we are trying to find a dense pattern in a sparse target. In the purple region, at the top left, every instance is satisfiable—we are looking for a sparse pattern in a dense target (which is easy, since we only have to preserve adjacency, not non-adjacency). The white band between the regions shows the location of the phase transition: here, roughly half the instances are satisfiable. (We discuss the black line below.)

On subsequent rows, we show the average number of search nodes used by the different algorithms. In general, satisfiable instances are easy, until very close to the phase transition. As we hit the phase transition and move into the unsatisfiable region, we see complexity increase. Finally, as we pass through the phase transition and move deeper into the unsatisfiable region, instances become easier again. This behaviour is largely solver-independent, although VF2 has a larger hard region than Glasgow or LAD. Thus, although we have moved away from a single control parameter, we still observe the easy-hard-easy pattern seen in many other NP-complete problems.

## 2.1 Locating the Phase Transition

We can approximately predict the location of the phase transition by calculating (with simplifications regarding rounding and independence) the expected number of solutions for given parameters. Since we are trying to find an *injective* mapping from a pattern $P = G(p, d_p)$ to a target $T = G(t, d_t)$, there are

$$t^{\underline{p}} = t \cdot (t-1) \cdot \ldots \cdot (t-p+1)$$

possible assignments of target vertices to pattern vertices. We expect the pattern to have $d_p \cdot \binom{p}{2}$ edges, so we obtain the probability of all of these edges being mapped to edges in the target by raising $d_t$ to this power, giving an expected number of solutions of

$$\langle Sol \rangle = t^{\underline{p}} \cdot {d_t}^{d_p \cdot \binom{p}{2}}.$$

This formula predicts a very sharp phase transition from $\langle Sol \rangle \ll 1$ to $\langle Sol \rangle \gg 1$, which may easily be located numer-

ically. We plot where this occurs using black lines in the first row of Figure 3.

This prediction is generally reasonably accurate, except that for very low and very high pattern densities, we overestimate the satisfiable region. This is due to variance: although an expected number of solutions much below one implies a high likelihood of unsatisfiability, it is not true that a high expected number of solutions implies that any particular instance is likely to be satisfiable. (Consider, for example, a sparse graph which has several isolated vertices. If one solution exists, other symmetric solutions can be obtained by permuting the isolated vertices. Thus although the expected number of solutions may be one, there cannot be exactly one solution.) A similar behaviour is seen with random constraint satisfaction problems [Smith and Dyer, 1996].

### 2.2 Variable and Value Ordering Heuristics

Various general principles have been considered when designing variable and value ordering heuristics for backtracking search algorithms—one of these is to try to maximise the expected number of solutions inside any subproblem considered during search [Gent *et al.*, 1996a]. This is usually done by cheaper surrogates, rather than direct calculation. When branching, both LAD and Glasgow pick a variable with fewest remaining values in its domain: doing this will generally reduce the first part of the $\langle Sol \rangle$ equation by as little as possible. When two or more domains are of equal size, LAD simply breaks ties lexicographically, whereas Glasgow will pick a variable corresponding to a pattern vertex of highest degree. This strategy was determined empirically, but could have been derived from the $\langle Sol \rangle$ formula: picking a pattern vertex of high degree will make the remaining pattern subgraph sparser, which will decrease the exponent in the second half of the formula, maximising the overall value. LAD does not apply a value ordering heuristic, but Glasgow does: it prefers target vertices of lowest degree. Again, this was determined empirically, but it has the effect of increasing $\langle Sol \rangle$ by increasing the remaining target density. The VF2 heuristics, in contrast, are based around preserving connectivity, which gives very little discrimination except on the sparsest of inputs.

## 3 Induced Subgraph Isomorphisms

In the first four rows of Figure 4 we repeat our experiments, finding induced isomorphisms. With a pattern of order 10, we get two independent phase transitions: the bottom right half of the plots resemble the non-induced results, and the top left half is close to a mirror image. The central satisfiable region, which is away from either phase transition, is computationally easy, but instances near the phase transition are hard.

For larger patterns of order 20 and 30, we have a large unsatisfiable region in the middle. Despite not being near either phase transition, instances in the centre remain computationally challenging. We also plot patterns of orders 14, 15 and 16, to show the transition between the two behaviours.

### 3.1 Predictions and Heuristics

To predict the location of the induced phase transition, we repeat the argument for locating the non-induced phase transition and additionally considering non-edges, to get an expected number of solutions of

$$\langle Sol \rangle = t_{\underline{}}^{p} \cdot {d_t}^{d_p \cdot \binom{p}{2}} \cdot (1 - d_t)^{(1-d_p) \cdot \binom{p}{2}}.$$

We plot this using black lines on the top row of Figure 4—again, our prediction is accurate except for very sparse or very dense patterns.

We might guess that degree-based heuristics would just not work for the induced problem: for any claim about the degree, the opposite will hold for the complement constraints. However, empirically, this is not the case: on the final row of Figure 4, we show whether it is better to use the original pattern and target as the input to the Glasgow algorithm, or to take the complements. (The only steps performed by the Glagsow algorithm which differ under taking the complements are the degree-based heuristics. LAD and VF2 are not symmetric in this way: LAD performs a filtering step using degree information, but does not consider the complement degree, and VF2 uses connectivity in the pattern graph.)

For patterns of order 10, it is always better to try to move towards the satisfiable region: if we are in the bottom right diagonal half, we are best retaining the original heuristics (which move us towards the top left), and if we are in the top left we should use the complement instead. This goes against a suggestion by Walsh [1998] that switching heuristics based upon an estimate of the solubility of the problem may offer good performance.

For larger patterns, more complex behaviour emerges. If we are in the intersection of the bottom half and the bottom right diagonal of the search space, we should always retain the original heuristic, and if we are in the intersection of the top half and the top left diagonal, we should always use the complements. This behaviour can be predicted by taking the partial derivatives of $\langle Sol \rangle$ in the $-p_d$ and $t_d$ directions. However, when inside the remaining two eighths of the parameter space, the partial derivatives of $\langle Sol \rangle$ disagree on which heuristic to use, and using directional derivatives is not enough to resolve the problem. A close observation of the data suggests that the actual location of the phase transition may be involved (and perhaps Walsh's suggestion applies only in these conditions). In any case, $\langle Sol \rangle$ is insufficient to explain the observed behaviour in these two eighths of the parameter space.

In practice, this is unlikely to be a problem: most real-world instances are extremely sparse and are usually easy, which perhaps explains the continuing popularity of VF2's connectivity-based heuristics [Carletti *et al.*, 2015]. In this situation, these experiments justify reusing the non-induced heuristics on induced problems.

### 3.2 Is the Central Region Genuinely Hard?

The region in the parameter space where both pattern and target have medium density is far from a phase transition, but nevertheless contains instances that are hard for all three solvers. We would like to know whether this is due to a weakness in current solvers (perhaps our solvers cannot reason about adjacency and non-adjacency simultaneously?), or whether instances in this region are inherently difficult to solve. Thus we repeat the induced experiments on smaller pattern and target graphs, using different solving techniques. Although these techniques are not competitive in absolute terms, we wish to

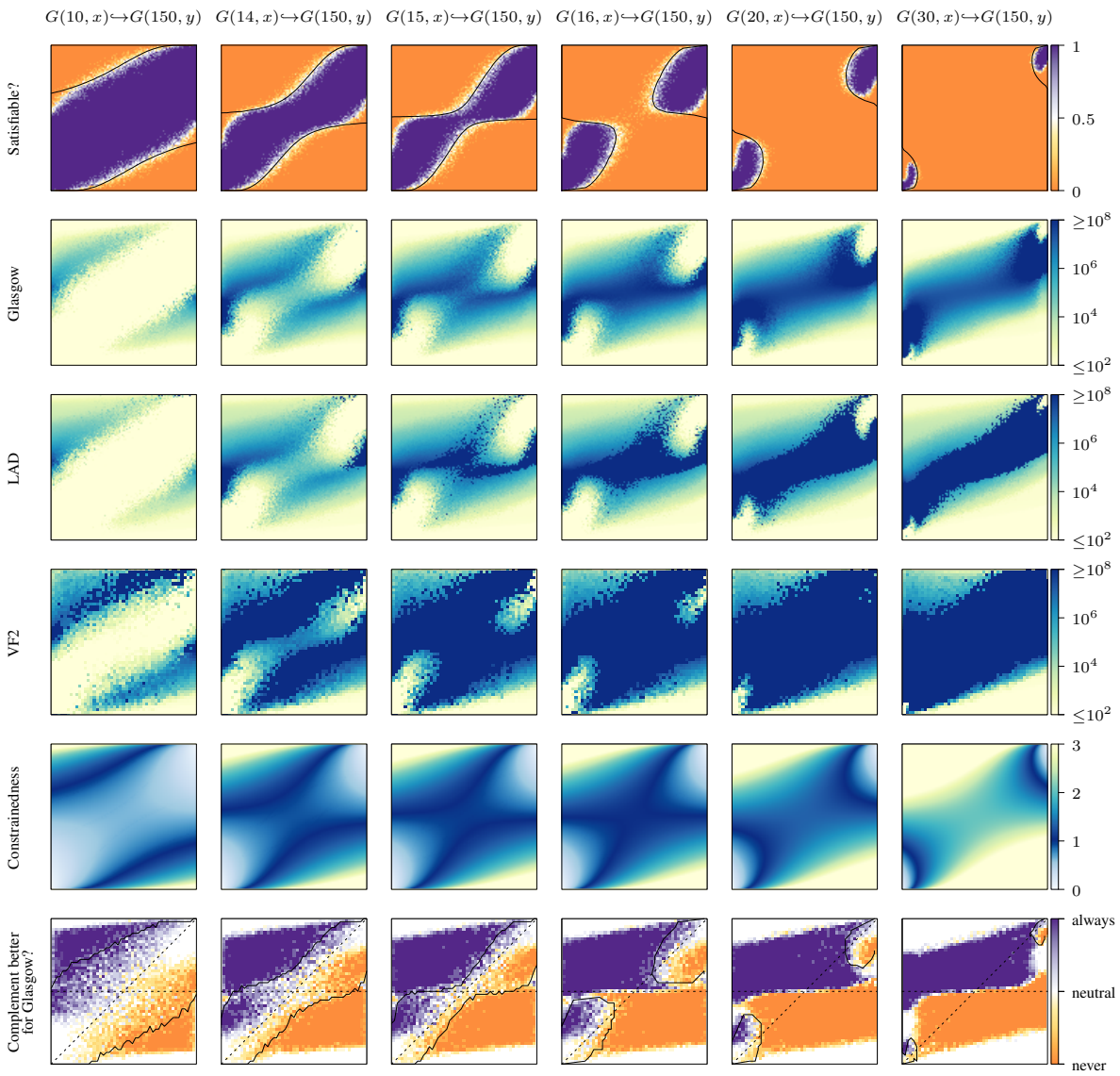


Figure 4: Behaviour of algorithms on the induced variant, shown in the style of Figure 3. The second, third and fourth rows show the number of search nodes used by the Glasgow, LAD and VF2 algorithms. The fifth row plots constrainedness: the darkest region is where $\kappa = 1$, and the lighter regions show where the problem is either over- or under-constrained. The final row shows when the Glasgow algorithm performs better when given the complements of the pattern and target graphs as inputs—the solid lines show the location of the phase transition, and the dotted lines are $t_d = 0.5$ and the $p_d = t_d$ diagonal.

see if the same pattern of behaviour occurs. The results are plotted in Figure 5.

The pseudo-boolean (PB) encoding is as follows. For each pattern vertex $v$ and each target vertex $w$, we have a binary variable which takes the value 1 if and only if $v$ is mapped to $w$. Constraints are added to ensure that each pattern vertex maps to exactly one target vertex, that each target vertex is mapped to by at most one pattern vertex, that adjacent vertices are mapped to adjacent vertices, and that non-adjacent vertices are mapped to non-adjacent vertices. We used the Clasp solver [Gebser *et al.*, 2011] version 3.1.3 to solve the pseudo-boolean instances. The instances that are hard for the Glasgow solver remain hard for the PB solver, including instances inside the central region, and the easy satisfiable instances remain easy. Similar results were seen with the Glucose SAT solver [Audemard and Simon, 2014] using a direct encoding of the cardinality constraints. (We also implemented an integer program encoding; the Gurobi solver was only able to solve some of the trivial satisfiable instances, and was almost never able to prove unsatisfiability within the time limit.)

The *association graph encoding* of a subgraph isomorphism problem (illustrated in Figure 1) is constructed by creating a new graph with a vertex for each pair $(p, t)$ of vertices from the pattern and target graphs respectively. There is an edge

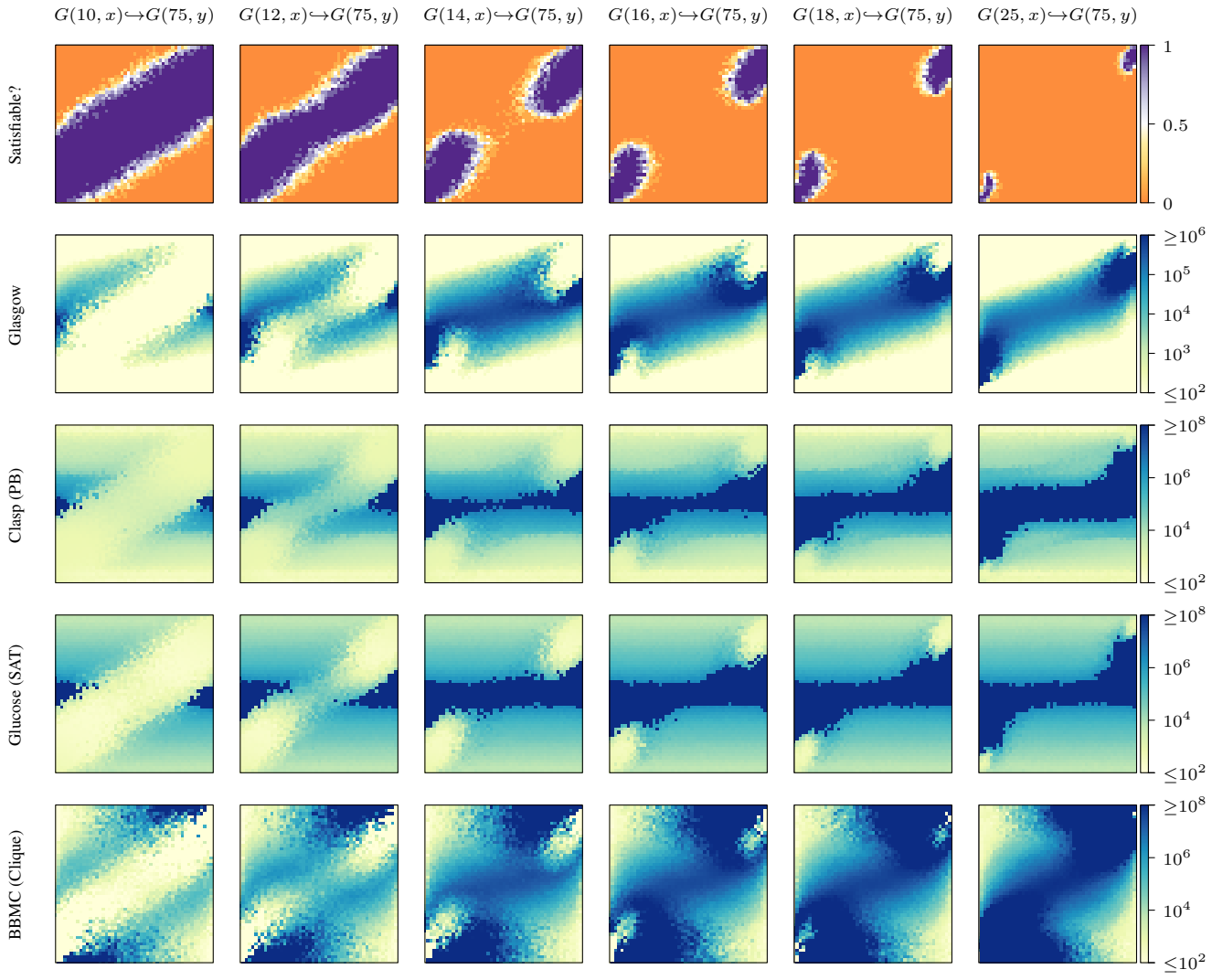


Figure 5: Behaviour of other solvers on the induced variant on smaller graphs, shown in the style of Figure 3. The second row shows the number of search nodes used by the Glasgow algorithm, the third and fourth rows show the number of decisions made by the pseudo-boolean and SAT solvers, and the final shows the number of search nodes used on the clique encoding.

between vertex $(p_1, t_1)$ and vertex $(p_2, t_2)$ if mapping $p_1$ to $t_1$ and $p_2$ to $t_2$ simultaneously is permitted, i.e. $p_1$ is adjacent to $p_2$ if and only if $t_1$ is adjacent to $t_2$. A clique of size equal to the order of the pattern graph exists in the association graph if and only if the problem is satisfiable [Levi, 1973]. We used this encoding with the BBMC clique algorithm [San Segundo *et al.*, 2011], which we implemented in C++. Usually BBMC solves the optimisation version of the clique problem; we adapted it to solve the decision problem by initialising the incumbent to be one less than the decision value, and allowing it to exit as soon as a clique with size equal to the decision value is encountered. Again, our results show that the instances in the central region remain hard, and additionally, some of the easy unsatisfiable instances become hard.

Together, these experiments suggest that the central region may be genuinely hard, despite not being near a phase transition. The clique results in particular rule out the hypothesis that subgraph isomorphism solvers only find this region hard due to not reasoning simultaneously about adjacency and non-adjacency, since the association graph encoding constraints consider compatibility rather than adjacency and non-adjacency.

### 3.3 Constrainedness

Constrainedness, denoted $\kappa$, is an alternative measure of difficulty designed to refine the phase transition concept, and to generalise hardness parameters across different combinatorial problems [Gent *et al.*, 1996b]. A problem with $\kappa < 1$ is said to be underconstrained, and is likely to be satisfiable; a problem with $\kappa > 1$ is overconstrained, and is likely to be unsatisfiable. Empirically, problems with $\kappa$ close to 1 are hard, and problems where $\kappa$ is very small or very large are usually

easy. By handling injectivity as a restriction on the size of the state space rather than as a constraint, we derive

$$\kappa = 1 - \frac{\log\left(t_{\underline{\ }}^{p} \cdot {d_t}^{d_p \cdot \binom{p}{2}} \cdot (1-d_t)^{(1-d_p)\cdot\binom{p}{2}}\right)}{\log t_{\underline{\ }}^{p}}$$

for induced isomorphisms, which we plot on the fifth row of Figure 4. We see that constrainedness predicts that the central region will still be relatively difficult for larger pattern graphs: although the problem is overconstrained, it is less overconstrained than in the regions the Glasgow and LAD solvers found easy. Thus it seems that rather than just being a unification of existing generalised heuristic techniques, constrainedness also gives a better predictor of difficulty than proximity to a phase transition—our method generates instances where constrainedness and "close to a phase transition" give very different predictions, and constrainedness gives the better prediction.

Unfortunately, constrainedness does not help us with heuristics: minimising constrainedness gives the same predictions as maximising the expected number of solutions.

## 4 Conclusion

We have shown how to generate small but hard instances for the non-induced and induced subgraph isomorphism problems, which will help offset the bias in existing datasets. For non-induced isomorphisms, behaviour was as in many other hard problems, but for induced isomorphisms we uncovered several interesting phenomena: there are hard instances far from a phase transition, constrainedness predicts this, and existing general techniques for designing heuristics do not work in certain portions of the parameter space.

The model we have proposed may be extended to graphs with labels on the vertices or edges, which arise naturally in many applications. Broadly speaking, randomly allocated labels make the problem easier by restricting the search space, although there are pathological cases where increasing the number of labels shifts an instance from the easy satisfiable region to being close to the phase transition. The model may also be extended to the (decision version of the) maximum common subgraph problem. The maximum clique approach is competitive for the maximum common subgraph problem, so it is interesting to observe that some of the easy portions of the search space become hard under this reduction—this suggests a possible unnecessary weakness of maximum clique algorithms on certain kinds of input. (Maximum clique algorithms use a greedy colouring as a bound, and a preliminary investigation suggests that the association graphs produced by this reduction can be very bad for greedy colourings in the same way that crown graphs are.)

In contrast, it is worth noting that this technique does *not* give a way of generating hard instances for graph isomorphism problems: the pattern graph must be substantially smaller than the target graph for independent pairs of randomly generated instances to give interesting behaviour.

In future work, we intend to repeat the experiments using other random models, including bounded, regular degree, and scale-free; regular degree graphs in particular foil existing degree-based heuristics. Similarly, we will look at alternatives to simple randomness for labelling strategies—for example, in chemical datasets, label sets tend to differ based upon the degrees of their corresponding vertices. We will also look at dynamic heuristics, and switching pattern and target heuristics independently. Finally, we intend to investigate whether variance can be calculated efficiently enough to give better predictions for very sparse or dense pattern graphs.


## Acknowledgements

The authors wish to thank Kitty Meeks and Craig Reilly for their comments.


## References


[Anton and Olson, 2009] Călin Anton and Lane Olson. Generating satisfiable sat instances using random subgraph isomorphism. In Yong Gao and Nathalie Japkowicz, editors, *Advances in Artificial Intelligence*, volume 5549 of *Lecture Notes in Computer Science*, pages 16–26. Springer Berlin Heidelberg, 2009.

[Audemard and Simon, 2014] Gilles Audemard and Laurent Simon. The Glucose SAT solver, 2014.

[Audemard *et al.*, 2014] Gilles Audemard, Christophe Lecoutre, Mouny Samy Modeliar, Gilles Goncalves, and Daniel Porumbel. Scoring-based neighborhood dominance for the subgraph isomorphism problem. In *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, pages 125–141, 2014.

[Battiti and Mascia, 2007] R. Battiti and F. Mascia. An algorithm portfolio for the sub-graph isomorphism problem. In *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics, International Workshop, SLS 2007, Brussels, Belgium, September 6-8, 2007, Proceedings*, volume 4638 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 2007.

[Carletti *et al.*, 2015] Vincenzo Carletti, Pasquale Foggia, and Mario Vento. *Graph-Based Representations in Pattern Recognition: 10th IAPR-TC-15 International Workshop, GbRPR 2015, Beijing, China, May 13-15, 2015. Proceedings*, chapter VF2 Plus: An Improved version of VF2 for Biological Graphs, pages 168–177. Springer International Publishing, Cham, 2015.

[Cheeseman *et al.*, 1991] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In John Mylopoulos and Raymond Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991*, pages 331–340. Morgan Kaufmann, 1991.

[Conte *et al.*, 2004] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(03):265–298, 2004.

[Cordella *et al.*, 2004] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10):1367–1372, 2004.

[Damiand *et al.*, 2011] Guillaume Damiand, Christine Solnon, Colin de la Higuera, Jean-Christophe Janodet, and Émilie Samuel. Polynomial algorithms for subisomorphism of $n$D open combinatorial maps. *Computer Vision and Image Understanding*, 115(7):996 – 1010, 2011. Special issue on Graph-Based Representations in Computer Vision.

[De Santo *et al.*, 2003] M. De Santo, P. Foggia, C. Sansone, and M. Vento. A large database of graphs and its use for benchmarking graph isomorphism algorithms. *Pattern Recogn. Lett.*, 24(8):10671079, May 2003.

[Gebser *et al.*, 2011] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam answer set solving collection. *AI Communications*, 24(2):107–124, 2011.

[Gent *et al.*, 1996a] Ian P. Gent, Ewan MacIntyre, Patrick Prosser, Barbara M. Smith, and Toby Walsh. An empirical study of dynamic variable ordering heuristics for the constraint satisfaction problem. In *Proceedings of the Second International Conference on Principles and Practice of Constraint Programming, Cambridge, Massachusetts, USA, August 19-22, 1996*, pages 179–193, 1996.

[Gent *et al.*, 1996b] Ian P. Gent, Ewan MacIntyre, Patrick Prosser, and Toby Walsh. The constrainedness of search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, August 4-8, 1996, Volume 1.*, pages 246–252, 1996.

[Giugno *et al.*, 2013] Rosalba Giugno, Vincenzo Bonnici, Nicola Bombieri, Alfredo Pulvirenti, Alfredo Ferro, and Dennis Shasha. GRAPES: A software for parallel searching on biological graphs targeting multi-core architectures. *PLoS ONE*, 8(10):e76911, 10 2013.

[Levi, 1973] G. Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *CALCOLO*, 9(4):341–352, 1973.

[Leyton-Brown *et al.*, 2014] Kevin Leyton-Brown, Holger H. Hoos, Frank Hutter, and Lin Xu. Understanding the empirical hardness of NP-complete problems. *Commun. ACM*, 57(5):98–107, May 2014.

[McCreesh and Prosser, 2015] Ciaran McCreesh and Patrick Prosser. A parallel, backjumping subgraph isomorphism algorithm using supplemental graphs. In Gilles Pesant, editor, *Principles and Practice of Constraint Programming*, volume 9255 of *Lecture Notes in Computer Science*, pages 295–312. Springer International Publishing, 2015.

[Mitchell *et al.*, 1992] David G. Mitchell, Bart Selman, and Hector J. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of the 10th National Conference on Artificial Intelligence. San Jose, CA, July 12-16, 1992.*, pages 459–465, 1992.

[San Segundo *et al.*, 2011] Pablo San Segundo, Diego Rodríguez-Losada, and Agustín Jiménez. An exact bit-parallel algorithm for the maximum clique problem. *Comput. Oper. Res.*, 38(2):571–581, February 2011.

[Smith and Dyer, 1996] Barbara M. Smith and Martin E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artif. Intell.*, 81(1-2):155–181, 1996.

[Smith and Grant, 1997] Barbara M. Smith and Stuart A. Grant. Modelling exceptionally hard constraint satisfaction problems. In Gert Smolka, editor, *Principles and Practice of Constraint Programming-CP97*, volume 1330 of *Lecture Notes in Computer Science*, pages 182–195. Springer Berlin Heidelberg, 1997.

[Solnon *et al.*, 2015] Christine Solnon, Guillaume Damiand, Colin de la Higuera, and Jean-Christophe Janodet. On the complexity of submap isomorphism and maximum common submap problems. *Pattern Recognition*, 48(2):302 – 316, 2015.

[Solnon, 2010] Christine Solnon. Alldifferent-based filtering for subgraph isomorphism. *Artif. Intell.*, 174(12-13):850–864, 2010.

[Walsh, 1998] Toby Walsh. The constrainedness knife-edge. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, July 26-30, 1998, Madison, Wisconsin, USA.*, pages 406–411, 1998.

[Zampelli *et al.*, 2010] S. Zampelli, Y. Deville, and C. Solnon. Solving subgraph isomorphism problems with constraint programming. *Constraints*, 15(3):327–353, 2010.