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

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.

\[
\langle \text{Sol} \rangle = t \cdot (t - 1) \cdot \ldots \cdot (t - p + 1) \cdot d_p^k(t),
\]

By considering when \(\langle \text{Sol} \rangle = 1\), 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.

Non-Induced Phase Transitions

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

\[
\langle \text{Sol} \rangle = t \cdot (t - 1) \cdot \ldots \cdot (t - p + 1) \cdot d_p^k(t) \cdot (1 - d_t)^{(p-1)(t-1)}.
\]

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 \(\langle \text{Sol} \rangle\) 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 eigths of the search space (and constrainedness does not help).

Future Work

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

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 on the expected number of solutions?

This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/K503058/1]. c.mccreesh.1@research.gla.ac.uk