Non-Induced Subgraph Isomorphism

\[
\begin{array}{c}
\begin{array}{c}
1 \\
2 \\
3 \\
4
\end{array} \\
\begin{array}{c}
1 \\
2 \\
3 \\
4 \\
5 \\
6
\end{array}
\end{array}
\]
Non-Induced Subgraph Isomorphism

Ciaran McCreesh, Patrick Prosser and James Trimble

Subgraph Isomorphism in Practice
The Algorithm

- Recursively build up a mapping from vertices of the pattern graph to vertices of the target graph.

- In constraint programming terms:
  - Forward-checking recursive search.
  - A variable for every pattern vertex.
  - Initially, each domain contains every target vertex.
  - After guessed assignments, infeasible values are eliminated from domains.
    - All-different constraint.
    - Adjacency constraints.
  - If we get a wipeout, we backtrack.
But wait! There’s more!

- Clever filtering at the top of search using neighbourhood degree sequences and paths, to reduce the initial values of domains.
- Pre-computed path count constraints, propagated like adjacency constraints during search.
- Bit-parallel implementation.
  - Weaker than the usual all-different propagator, but much faster.
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.
Is It Any Good?

Number of Instances Solved vs. Runtime (ms)

- Somewhere Exotic (not yet written)
- ESA 2018 (not yet rejected)
- LION 2016 (a)
- CP 2015
- LION 2016 (b)
- AIJ 2010
Search Order

- Variable ordering (i.e. pattern vertices): smallest domain first, tie-breaking on highest degree.
- Value ordering (i.e. target vertices): highest degree to lowest.
Hand-Wavy Theoretical Justification

- Maximise the expected number of solutions during search?
- If $P = G(p, q)$ and $T = G(t, u)$,

$$\langle \text{Sol} \rangle = t \cdot (t - 1) \cdot \ldots \cdot (t - p + 1) \cdot u^{q \cdot \binom{p}{2}}$$

- Smallest domain first keeps remaining domain sizes large.
- High pattern degree makes the remaining pattern subgraph sparser, reducing $q$.
- High target degree leaves as many vertices as possible available for future use, making $u$ larger.
Sanity Check

Number of Sat Instances Solved vs. Runtime (ms) for different degrees of difficulty:
- Degree
- Random
- Anti

Graph shows the number of satisfiable instances solved against runtime, with three lines representing different difficulty levels.
Sanity Check

Number of Unsat Instances Solved vs. Runtime (ms)
Phase Transitions

\[ G(10, x) \rightarrow G(150, y) \]

\[ G(20, x) \rightarrow G(150, y) \]

\[ G(30, x) \rightarrow G(150, y) \]
Incidentally, Induced is Much More Complicated

\[ G(10, x) \leftrightarrow G(150, y) \quad G(14, x) \leftrightarrow G(150, y) \quad G(16, x) \leftrightarrow G(150, y) \quad G(20, x) \leftrightarrow G(150, y) \]
However...

- Degree spread is low.
- We commit extremely heavily to the first branching choice, which is probably wrong.
Restarts

- Run search for a bit, and if we don’t find a solution, restart.
- Count number of backtracks, restart using the Luby sequence (with a magic constant multiplier).
  - 1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, …
- Obviously, something needs to change when we restart.
  - First attempt: random value-ordering heuristic.
Restarts

Ciaran McCreesh, Patrick Prosser and James Trimble

Subgraph Isomorphism in Practice
Restarts

Random + Restarts Search Time (ms)

Degree Search Time (ms)

Mesh sat
LV sat
Phase sat
Rand sat
Other sat
Any unsat
Nogoods

- Whenever we restart, post new constraints eliminating parts of the search space already explored.
- Potentially exponentially many constraints.
- But they are all in the form
  \[(d_1 = v_1) \land (d_2 = v_2) \land \ldots \land (d_n = v_n) \rightarrow \bot.\]

- Use two watched literals to propagate in \(O(1)\)ish time.
  - Basic idea: clauses only propagate when exactly one \((d_i = v_i)\) literal has not been set to true.
  - Watch two literals per clause that have not been set to true.
  - When unit propagating, only look at clauses with a watch corresponding to the assignment made.
  - Either find a new literal to watch, or propagate.
Nogoods

Number of Sat Instances Solved
Runtime (ms)
Random, nogoods
Random, restarts
Degree
Random
Anti

Ciaran McCreesh, Patrick Prosser and James Trimble
Subgraph Isomorphism in Practice
Nogoods

Random, Nogoods Search Time (ms)
Degree Search Time (ms)
Mesh sat
LV sat
Phase sat
Rand sat
Other sat
Any unsat

Ciaran McCreesh, Patrick Prosser and James Trimble
Subgraph Isomorphism in Practice
Biased Value-Ordering

- Select a vertex $v'$ from the chosen domain $D_v$ with probability
  \[ p(v') = \frac{2^{\deg(v')}}{\sum_{w \in D_v} 2^{\deg(w)}}. \]

- Looks a lot like \textit{softmax}, which uses base $e$. 
Biased Value-Ordering

Number of SAT Instances Solved

Runtime (ms)

Degree

Biased
Random
Anti

Ciaran McCreesh, Patrick Prosser and James Trimble

Subgraph Isomorphism in Practice
Biased Value-Ordering

![Graph showing Biased Search Time vs Degree Search Time with different markers for Mesh sat, LV sat, Phase sat, Rand sat, Other sat, and Any unsat.](image)
Biased Value-Ordering with Restarts and Nogoods

![Graph showing the number of SAT instances solved versus runtime. The x-axis represents the number of instances solved, and the y-axis represents the runtime (in milliseconds). There are four lines on the graph, each representing a different strategy: Biased, nogoods; Random, nogoods; Biased Degree; and Random Anti. The Biased, nogoods line is the highest, followed by Random, nogoods, Biased Degree, and Random Anti. The graph also shows the number of instances solved on a logarithmic scale.]
Biased Value-Ordering with Restarts and Nogoods

![Graph showing Biased, Nogoods Search Time versus Degree Search Time (ms)]

- Mesh sat
- LV sat
- Phase sat
- Rand sat
- Other sat
- Any unsat

Ciaran McCreesh, Patrick Prosser and James Trimble
Ongoing Work

- Is this form of search more broadly applicable?
- Specialisations, like clique, and generalisations, like maximum common subgraph.
- Parallelism.
- Subgraphs modulo theories.
- Algorithm engineering.