University of St Andrews

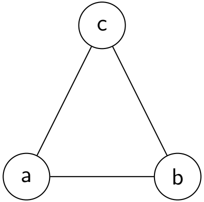
Symmetry Breaking in the Subgraph Isomorphism Problem

Joseph Loughney, *Ruth Hoffmann*, Mun See Chang and Ciaran McCreesh

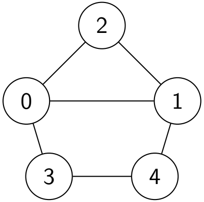
WHOOPS2025, EuroProofNet Symposium, Université Paris-Saclay, Institute Pascal

What is SIP?

Ciaran has already covered this.



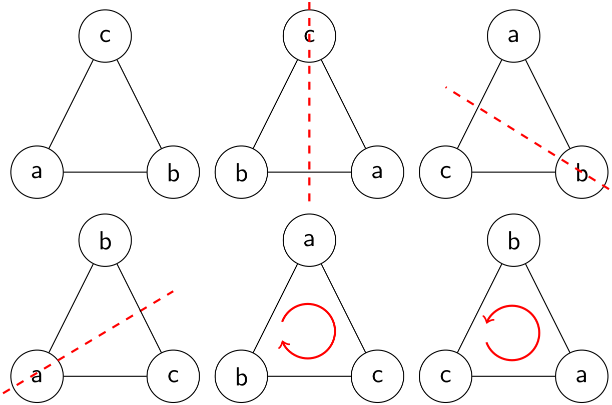
Pattern Graph



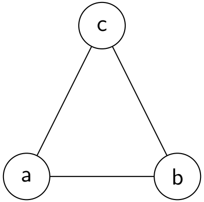
Target Graph

Symmetries of a Graph

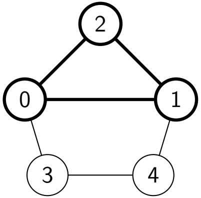
Markus has already covered this.



Symmetries in SIP



Pattern Graph



Target Graph

$$(a \mapsto 0, b \mapsto 1, c \mapsto 2), (a \mapsto 0, b \mapsto 2, c \mapsto 1),$$
$$(a \mapsto 1, b \mapsto 0, c \mapsto 2), (a \mapsto 1, b \mapsto 2, c \mapsto 0),$$
$$(a \mapsto 2, b \mapsto 0, c \mapsto 1), (a \mapsto 2, b \mapsto 1, c \mapsto 0)$$

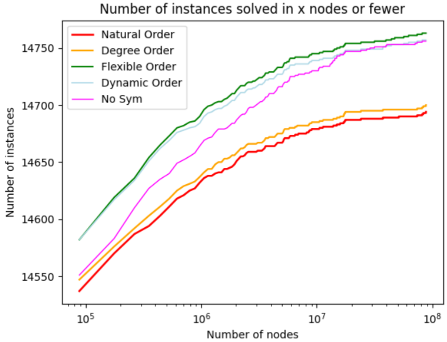
Different SB Approaches

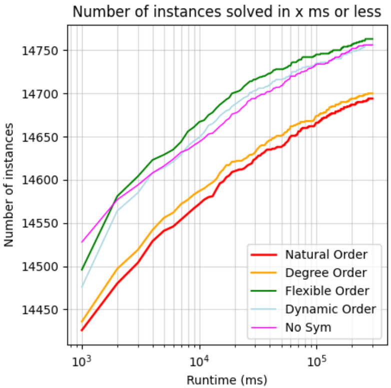
We have the following strategies:

- **Fixed, arbitrary** – pick an order at random before search. (Static, Markus covered this more or less)
- **Fixed, informed** – pick an order using the solver's search heuristics to try to approximate search order before search. (Static, Markus covered this more or less)
- **Flexible** – construct and refine an order in which variables/values are encountered.
- **Dynamic** – extend and retract orders according to the current search tree branch.

Progressing down the list yields more symmetry breaking, but more overhead.

Initial Results, on Pattern Graphs





Stabiliser Chain

All (our) SB approaches boil down to the same thing.

Use the information generated when finding a (strong) generating set of the automorphism group of the pattern/target graph.

- Which is (most efficiently) done using the Schreier-Sims algorithm which creates a stabiliser chain.
- And has been (very optimised for graphs) implemented by Markus Anders as part of `dejavu`. `https://automorphisms.org`

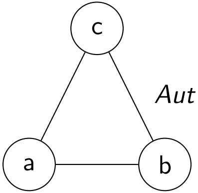
How can the Stabiliser Chain help with SB?

In general the stabiliser chain gives us two things:

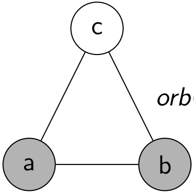
1. The orbits of the automorphism group (i.e. a set of equivalence classes).
2. Whenever we stabilise a vertex, Schreier-Sims gives us a set of permutations (the transversals).

Both give us a way of defining an order over the vertices of the graph.

Generating Constraints

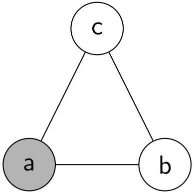


$$Aut(G) = \langle (a\ b\ c), (a\ b) \rangle$$

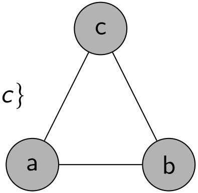


$$\beta_2 = b,$$
$$orb(b)|_{\beta_1 = a} = \{b, c\}$$
$$\implies b < c$$

$$\beta_1 = a,$$
$$orb(a) = \{a, b, c\}$$
$$\implies a < b, a < c$$



$$orb(c)|_{\beta_1 = a \wedge \beta_2 = b} = \{c\}$$



What do we have? Decision Problem

Mathematical proofs of validity in the decision problem when symmetry breaking using

- Pattern orbits
- Target orbits
- Pattern transversals
- Target transversals
- We *cannot* do pattern orbits + target orbits simultaneously
- Pattern transversals + target transversals simultaneously
- Pattern orbits + target transversals simultaneously
- Pattern transversals + target orbits simultaneously (this one might be computationally heavy)

All these can be done Statically, Flexibly, or Dynamically.

What do we have? Counting Problem

Mathematical proofs of validity in the counting problem when symmetry breaking using

- Pattern orbits
- Target orbits
- We *cannot* use pattern orbits + target orbits

All these can be done Statically, Flexibly, or Dynamically.
Currently working on transversals and doing both types of SB. (both the proofs and implementation)

What did we struggle with?

Not all of these can be solved using Proof Logging. We struggled through the implementation of our SB techniques

- Conflicting "Order" Constraints
- Conflicting with Propagators
- Undo-ing constraints

Upcoming Issues, Counting

- Overcounting
- Undercounting
- Constraint tracking

Decision Challenges for Proof Logging

Decision problem

- Fixed, arbitrary/informed, not really a challenge?
- Flexible, maybe a little challenge but ... not really?
- Dynamic, since we are adding/removing constraints on the fly... Problem?

Counting Challenges for Proof Logging

Counting Problem

- Fixed, arbitrary/informed, we know how to retrieve the count from the beginning.
- Flexible, we will be “building” the count retrieval during search and settle on it, proof?
- Dynamic, we are tracking which symmetric solutions are being pruned and which have been accounted for.

Finally

This work is undertaken by Joseph Loughney.

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

GSS with Symmetries (work in progress)
`https://github.com/ciaranm/glasgow-subgraph-solver/tree/dynamic-dejavu`

University of St Andrews

Thank you!

ruthhoffmann

https://www.st-andrews.ac.uk/computer-science/people/rh347

rh347@st-andrews.ac.uk