The Quest for Perfect and Compact Symmetry Breaking for Graph Problems

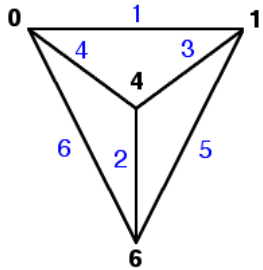
Marijn J.H. Heule

THE UNIVERSITY OF TEXAS AT AUSTIN

SYNASC September 25, 2016

Satisfiability (SAT) solving has many applications...

formal verification

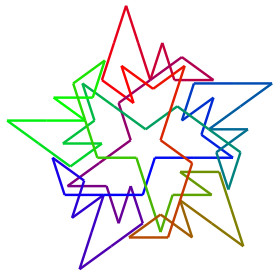


graph theory

bioinformatics

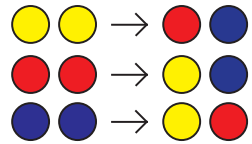
train safety

planning

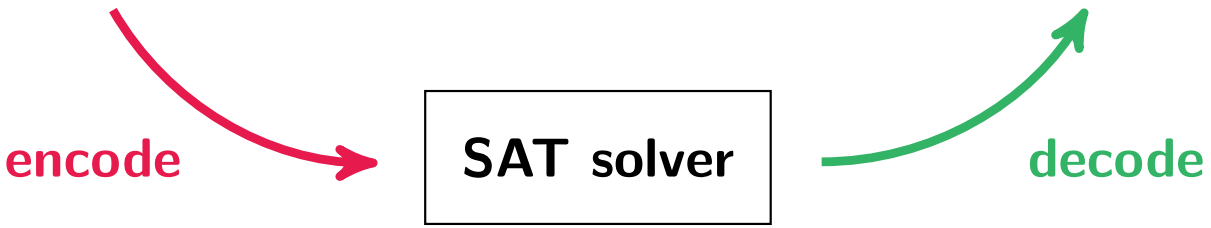


combinatorics

cryptography



rewrite termination



..., but SAT solving may struggle in the presence of symmetries

Breaking Graph Symmetries

Break Symmetries for Graph Existence Problems

A graph existence problem asks whether there exists a undirected graph with a certain property. For example, does every graph of six vertices have a clique or a co-clique of size 3? (Known as Ramsey number 3)

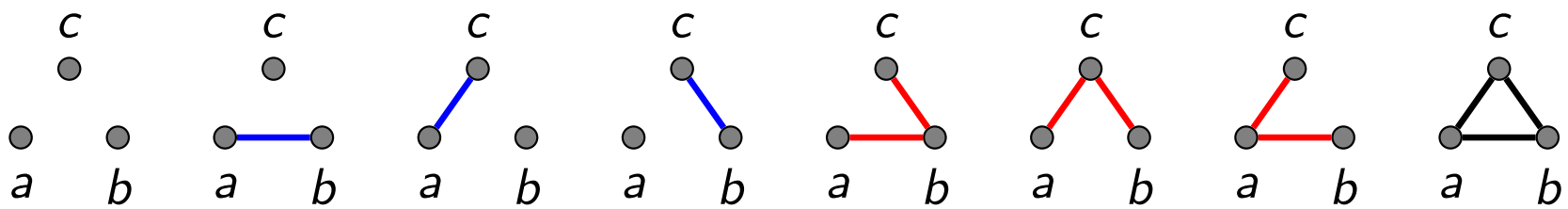
Graph existence problems are hard for SAT solvers due to the symmetries.

Break Symmetries for Graph Existence Problems

A graph existence problem asks whether there exists a undirected graph with a certain property. For example, does every graph of six vertices have a clique or a co-clique of size 3? (Known as Ramsey number 3)

Graph existence problems are hard for SAT solvers due to the symmetries.

Consider all graphs with three vertices:



We can perfectly break all symmetries by eliminating all but one graph from each isomorphism class. For example, eliminating graphs 3 to 6:

$$(ab \vee \overline{ac} \vee bc) \wedge (ab \vee ac \vee \overline{bc}) \wedge (\overline{ab} \vee ac \vee \overline{bc}) \wedge (ab \vee \overline{ac} \vee \overline{bc})$$

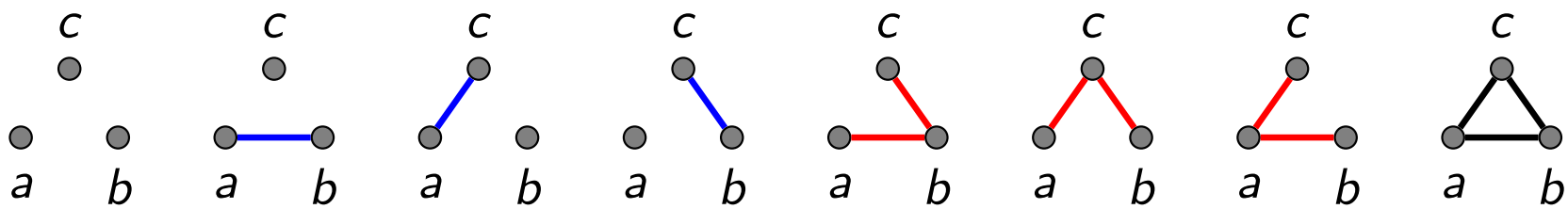
This can be simplified to $(ab \vee \overline{ac}) \wedge (ac \vee \overline{bc})$.

Break Symmetries for Graph Existence Problems

A graph existence problem asks whether there exists a undirected graph with a certain property. For example, does every graph of six vertices have a clique or a co-clique of size 3? (Known as Ramsey number 3)

Graph existence problems are hard for SAT solvers due to the symmetries.

Consider all graphs with three vertices:



We can perfectly break all symmetries by eliminating all but one graph from each isomorphism class. For example, eliminating graphs 3 to 6:

$$(ab \vee \overline{ac} \vee bc) \wedge (ab \vee ac \vee \overline{bc}) \wedge (\overline{ab} \vee ac \vee \overline{bc}) \wedge (ab \vee \overline{ac} \vee \overline{bc})$$

This can be simplified to $(ab \vee \overline{ac}) \wedge (ac \vee \overline{bc})$.

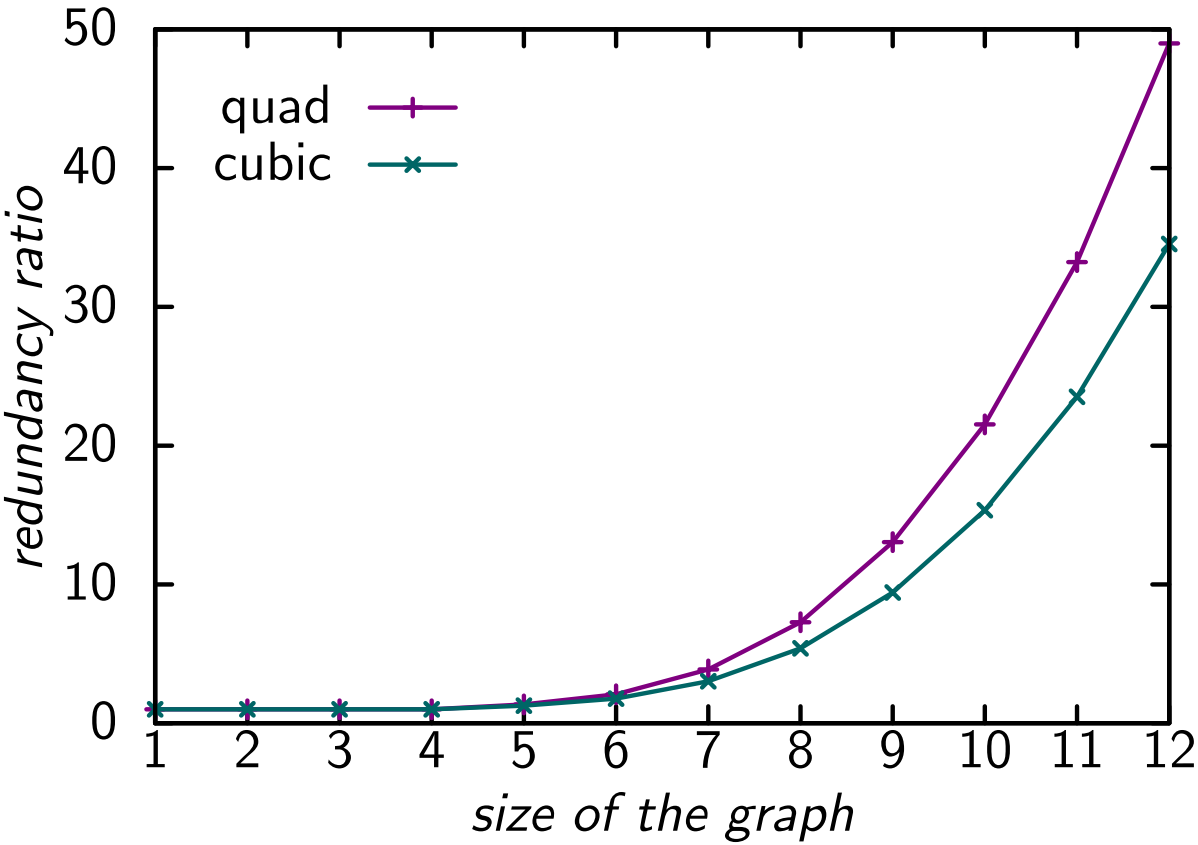
What size are the most compact perfect symmetry-breaking predicates?

Existing Techniques Break Symmetries Partially

Existing symmetry-breaking methods constrain the adjacency matrix:

quad Row i less than or equal to row $i+1$, while ignoring columns i, $i+1$

cubic Row i less than or equal to row j $(i < j)$, while ignoring columns i, j



Redundancy ratio: average number of graphs per isomorphism class

Computing Compact & Perfect Symmetry Breaking

Logic Minimization Method

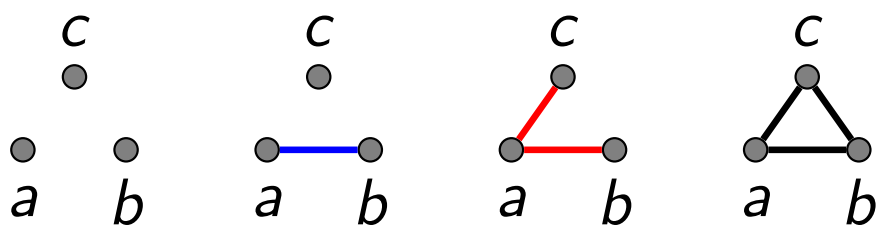
Perfect isolator: a perfect symmetry-breaking predicate

Compute a perfect isolator as follows:

1. Choose a canonical set of graphs, i.e., exactly one graph out of each isomorphism class;
2. Convert the canonical set into clauses (Tseitin encoding);
3. Reduce the size of the clauses via logic minimization.

For example for graphs with three vertices:

1. Canon:



2. Tseitin encoding results in a formula with 13 clauses (independent on canon).
3. Can be reduced to two clauses (dependent on canon).

Logic Minimization Sizes and Runtimes

Several tools exist to generate a canonical set:

- NAUTY by by Brendan McKay (1981)
- BLISS by Tommi Junttila and Petteri Kaski (2007)

Several tools exist to minimize a given logical formula:

- ESPRESSO by by Robert Brayton (1984)
- BICA by Alexey Ignatiev (2015)

Best results with NAUTY and BICA (size in cubes / clauses):

k	2	3	4	5	6	7	8
$\lvert P_{\text{DNF}}\rvert$	2	4	11	34	156	1,044	12,346
$\lvert P_{\text{CNF}}\rvert$	3	13	67	341	2,341	21,925	345,689
$\lvert P_{\text{simp}}\rvert$	0	2	9	24	77	311	$> 1,839$

SAT Solving Method

The prior method required a canonical set as input. However, the number of choices for the canonical set is exponential. Only some choices may be reducible to a compact predicate.

As an alternative approach, we translate the problem into SAT:

- Formula $F_{k,m}$ expresses the SAT encoding of the existence of a perfect isolator for k vertices using m clauses.
- All m clauses are satisfied by graphs in the canonical set;
- Each non-canonical graph falsifies at least one clause;
- These formulas are huge: $\mathcal{O}(2^{|E|}m|E|)$ with $|E| = \frac{k^2-k}{2}$.

Using the SAT approach, the optimal perfect symmetry breaking for graphs of size k can be computed: Find m such that $F_{k,m-1}$ is unsatisfiable, while $F_{k,m}$ is satisfiable.

SAT Solving Sizes and Runtimes

Formula $F_{k,m}$ expresses the SAT encoding of the existence of an isolator for k vertices using m clauses.

Two top-tier solvers: `glucose` (G) and `treengeling` (T)

formula	*result*	*variables*	*clauses*	*best runtime**
$F_{4,6}$	UNSAT	756	2,458	0.18 (G)
$F_{4,7}$	SAT	861	2,827	0.01 (G)
$F_{5,11}$	UNSAT	14,480	54,756	3,510.36 (T)
$F_{5,12}$	SAT	15,609	59,281	102.69 (G)

* Runtimes are in wall clock seconds on a quad core Intel Xeon E31280 CPU.

All formulas with $k \geq 6$ appeared too hard: i.e, unsolvable in 24 hours using a parallel solver running on 24 cores.

Random Probing Method

Logic minimization results in large perfect isolators, as existing canonicalization algorithms produce "poor" canonical sets.

The SAT method results in optimal isolators, but doesn't scale.

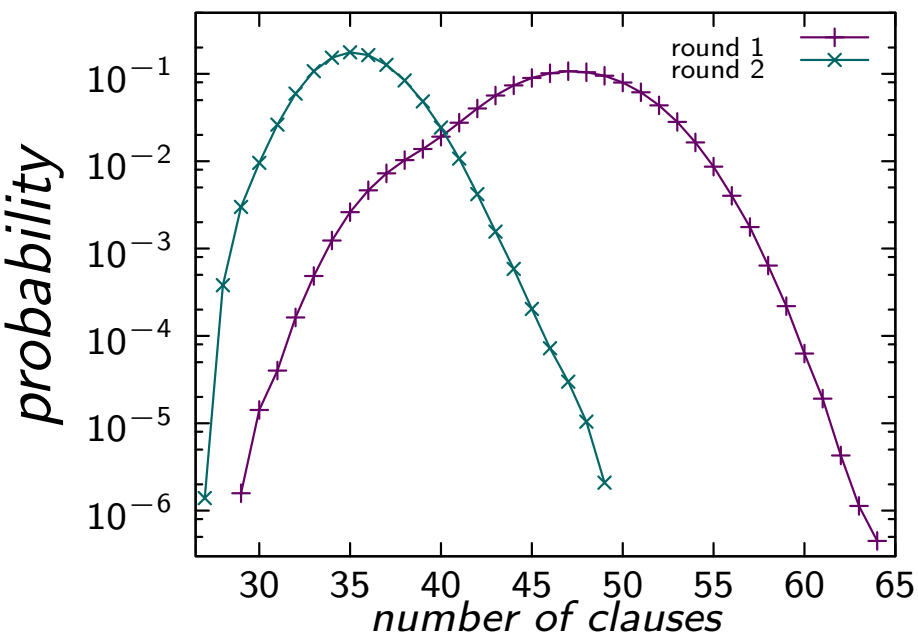
Our third approach is based on random probing:

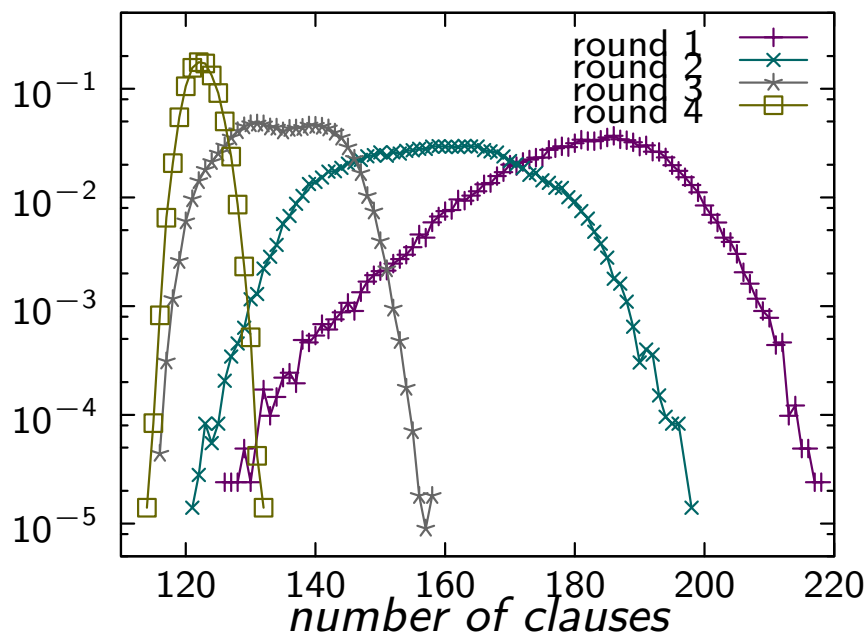
1. All graphs of size k are active and the isolator is empty.
2. Rank all potential clauses that can be added to the isolator. The more active graphs that are falsified by a clause, the higher its rank. Ties are broken randomly.
3. Randomly add a single clause to the isolator with probability $P(r) = 2^{-r}$, with r being the clause rank.
4. Terminate if the isolator is perfect. Otherwise to go 2.

Random Probing Sizes and Runtimes

Random probing can be improved by running multiple rounds:

- In round $i+1$ we pick the smallest isolators of round i and forced the first $10i$ clauses from those isolators.
- Below two probability plots: (left) the results of 2 rounds on $n = 6$ with 400,000 probes per round, and (right) the results of 4 rounds on $n = 7$ with 80,000 probes per round.





Results

Optimal Isolators in CNF

Variable xy denotes whether an edge from node x to y exists.
For example $bc = 0$ means there is no edge from node b to c.

$$P_3 := (ab \vee \overline{bc}) \wedge (bc \vee \overline{ac})$$

$$P_4 := (ad \vee \overline{bd}) \wedge (bd \vee \overline{ac}) \wedge (cd \vee \overline{bc}) \wedge (ab \vee \overline{bc}) \wedge (bc \vee \overline{ac}) \wedge (ab \vee bd \vee \overline{cd}) \wedge (bc \vee bd \vee \overline{ad})$$

$$P_5 := (ad \vee \overline{bd}) \wedge (bd \vee \overline{ac}) \wedge (cd \vee \overline{bc}) \wedge (bc \vee \overline{ad}) \wedge (ae \vee \overline{ce}) \wedge (be \vee \overline{ae}) \wedge (ab \vee bd \vee \overline{cd}) \wedge (ae \vee de \vee \overline{be}) \wedge (ad \vee ce \vee \overline{de}) \wedge (ab \vee \overline{cd} \vee \overline{de}) \wedge (ac \vee \overline{ad} \vee \overline{ce}) \wedge (ce \vee \overline{ab} \vee \overline{ae} \vee \overline{bc})$$

Optimal Isolators in CNF

Variable xy denotes whether an edge from node x to y exists.
For example $bc = 0$ means there is no edge from node b to c.

$$P_3 := (ab \vee \overline{bc}) \wedge (bc \vee \overline{ac})$$

$$P_4 := (ad \vee \overline{bd}) \wedge (bd \vee \overline{ac}) \wedge (cd \vee \overline{bc}) \wedge (ab \vee \overline{bc}) \wedge (bc \vee \overline{ac}) \wedge (ab \vee bd \vee \overline{cd}) \wedge (bc \vee bd \vee \overline{ad})$$

$$P_5 := (ad \vee \overline{bd}) \wedge (bd \vee \overline{ac}) \wedge (cd \vee \overline{bc}) \wedge (bc \vee \overline{ad}) \wedge (ae \vee \overline{ce}) \wedge (be \vee \overline{ae}) \wedge (ab \vee bd \vee \overline{cd}) \wedge (ae \vee de \vee \overline{be}) \wedge (ad \vee ce \vee \overline{de}) \wedge (ab \vee \overline{cd} \vee \overline{de}) \wedge (ac \vee \overline{ad} \vee \overline{ce}) \wedge (ce \vee \overline{ab} \vee \overline{ae} \vee \overline{bc})$$

Optimal Isolators in CNF

Variable xy denotes whether an edge from node x to y exists.
For example $bc = 0$ means there is no edge from node b to c.

$$P_3 := (ab \vee \overline{bc}) \wedge (bc \vee \overline{ac})$$

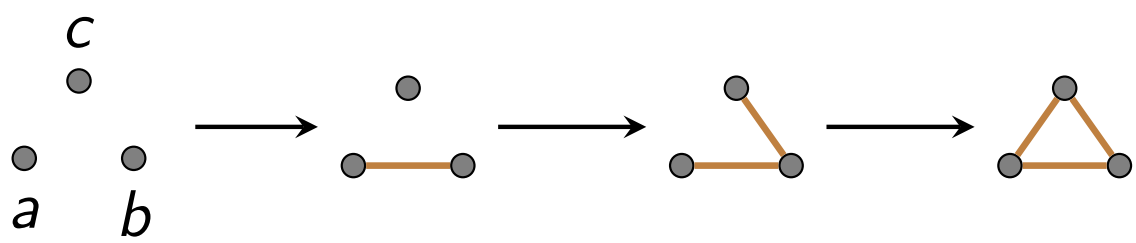
$$P_4 := (ad \vee \overline{bd}) \wedge (bd \vee \overline{ac}) \wedge (cd \vee \overline{bc}) \wedge (ab \vee \overline{bc}) \wedge (bc \vee \overline{ac}) \wedge (ab \vee bd \vee \overline{cd}) \wedge (bc \vee bd \vee \overline{ad})$$

$$P_5 := (ad \vee \overline{bd}) \wedge (bd \vee \overline{ac}) \wedge (cd \vee \overline{bc}) \wedge (bc \vee \overline{ad}) \wedge (ae \vee \overline{ce}) \wedge (be \vee \overline{ae}) \wedge (ab \vee bd \vee \overline{cd}) \wedge (ae \vee de \vee \overline{be}) \wedge (ad \vee ce \vee \overline{de}) \wedge (ab \vee \overline{cd} \vee \overline{de}) \wedge (ac \vee \overline{ad} \vee \overline{ce}) \wedge (ce \vee \overline{ab} \vee \overline{ae} \vee \overline{bc})$$

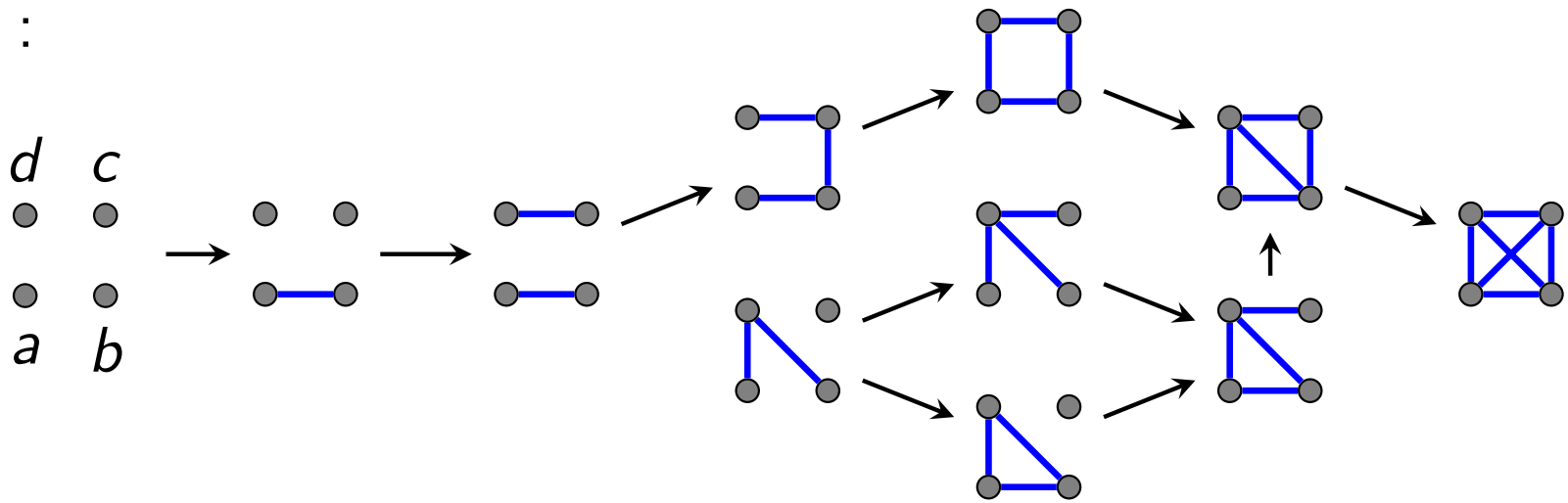
Visualization of Optimal Isolators for $n \in \{3, 4\}$

Two canonical graphs are connected with an arc if they differ in exactly one edge. The arrow points from the canonical graph without the edge to the one with the edge.

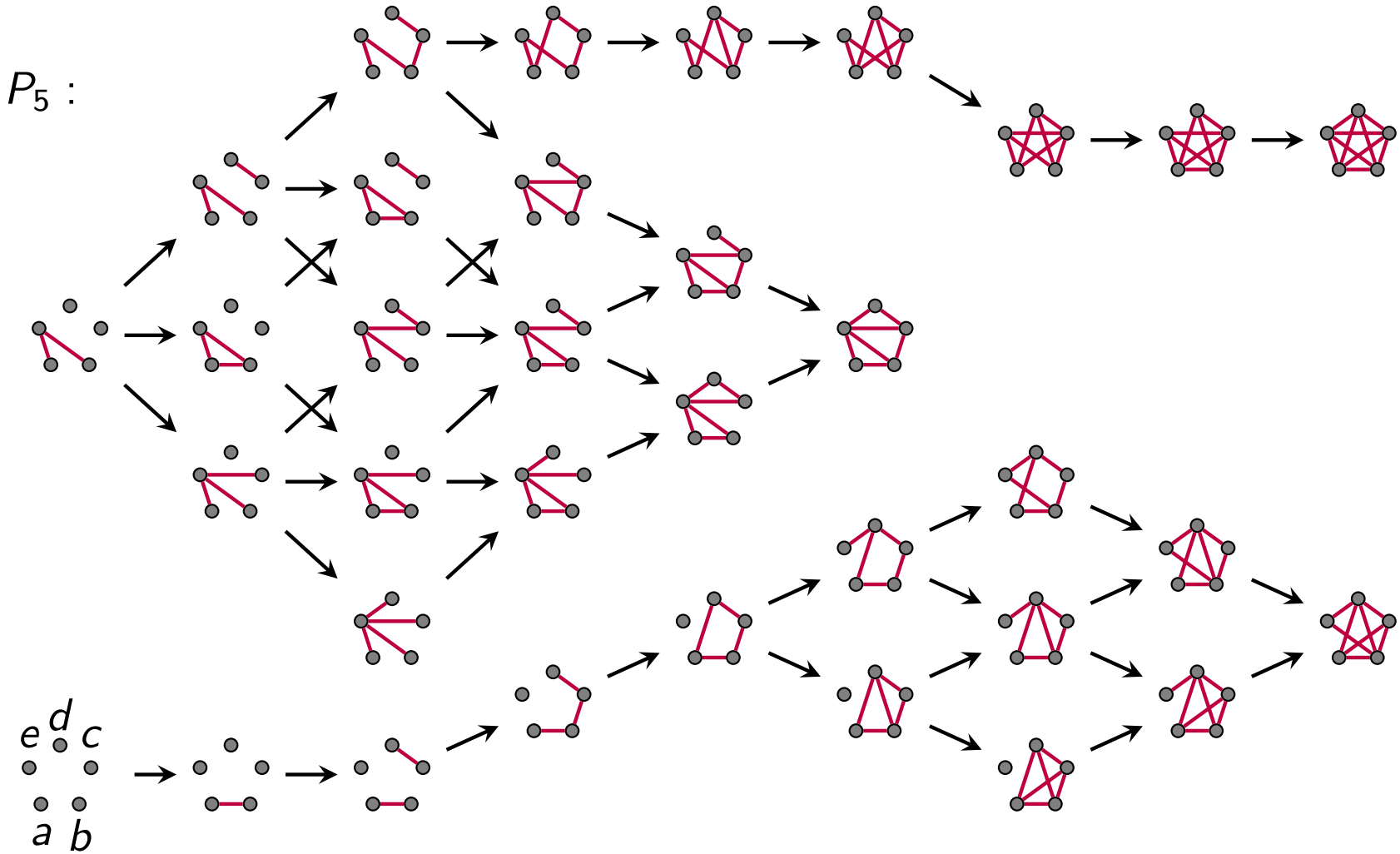
P_3 :



P_4 :



Visualization of Optimal Isolator for $n = 5$



Conclusions and Future Work

Conclusions and Future Work

Conclusions:

- We presented three methods to compute perfect symmetry-breaking predicated for graph problems.
- Optimal isolators are compact, at least for small graphs.
- Existing canonical label algorithms do not allow the construction of small isolators.

Conclusions and Future Work

Conclusions:

- We presented three methods to compute perfect symmetry-breaking predicated for graph problems.
- Optimal isolators are compact, at least for small graphs.
- Existing canonical label algorithms do not allow the construction of small isolators.

Future work:

- How to compute optimal isolators for medium graphs?
- How to construct compact isolators for large graphs?
- Can graphs symmetries be perfectly broken using polynomial-sized predicates?

Conclusions and Future Work

Conclusions:

- We presented three methods to compute perfect symmetry-breaking predicated for graph problems.
- Optimal isolators are compact, at least for small graphs.
- Existing canonical label algorithms do not allow the construction of small isolators.

Future work:

- How to compute optimal isolators for medium graphs?
- How to construct compact isolators for large graphs?
- Can graphs symmetries be perfectly broken using polynomial-sized predicates?

Thanks!