Incremental SAT-Based Enumeration of Solutions to the Yang-Baxter Equation

Daimy Van Caudenberg, Bart Bogaerts, Leandro Vendramin
KU Leuven, Vrije Universiteit Brussel

TACAS 2025

KU LEUVEN

THE YANG-BAXTER EQUATION

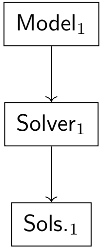
- Originally introduced in the context of statistical [Yan67] and quantum mechanics [Bax72].
- Has known applications in knot theory, quantum computing etc.
- The original equation is formulated over vector spaces, but a discrete version [Dri92] of the equation was introduced as well.
- We limit our focus to a subset of solutions to this discrete equation:
  - These solutions naturally have groups acting on them.
  - Tools from ring theory, group theory... can be used to analyze them.
  - They have connections with other topics, such as Hopf–Galois structures and knot theory.
- These specific solutions can be studied using an equivalent mathematical structure: non-degenerate cycle sets.

ENUMERATING SOLUTIONS

- Enumerating all such solutions to the YBE is still an open problem!
- A database of solutions could be used to:
  - allow for experimentation which can reveal patterns that provide deeper insights
  - find intriguing examples of algebraic structures to study
  - find counterexamples to previous conjectures

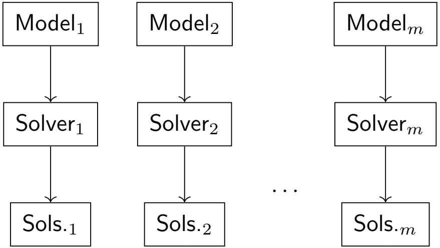
ENUMERATING CYCLE SETS

- As done by [AMV22]
- Model cycle set constraints.
- Add static symmetry breaking constraints.
- Enumerate all solutions.



ENUMERATING CYCLE SETS

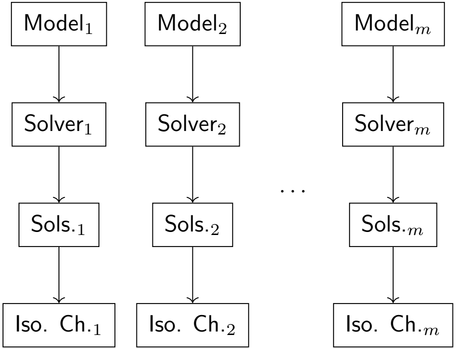
- As done by [AMV22]
- Model cycle set constraints.
- Partition the problem by fixing diagonals.
  - This decreases the search space per problem from $(n^2)^n$ to $(n^2 - n)^{(n-1)}$.
  - This allows to parallelize the search.
- Add static symmetry breaking constraints.



- Enumerate all solutions.

ENUMERATING CYCLE SETS

- As done by [AMV22]
- Model cycle set constraints.
- Partition the problem by fixing diagonals.
  - This decreases the search space per problem from $(n^2)^n$ to $(n^2 - n)^{(n-1)}$.
  - This allows to parallelize the search.
- Add static symmetry breaking constraints.
  - Complete symmetry breaking is unrealistic because of its encoding size...
- Enumerate all solutions.
- Perform final isomorphism check.

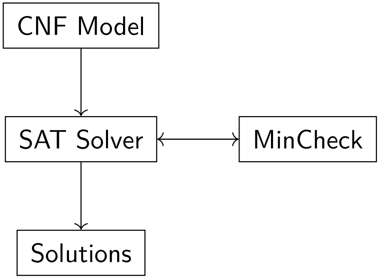


SAT MODULO SYMMETRIES [KS21]

- Main goal:
  - Enumerate satisfying assignments of Boolean formula up to isomorphism.
  - First used to enumerate graphs with certain interesting properties.
- Core idea:
  1. Model the mathematical problem at hand using propositional logic.
  2. Force a SAT solver to generate only non-isomorphic solutions during the search.

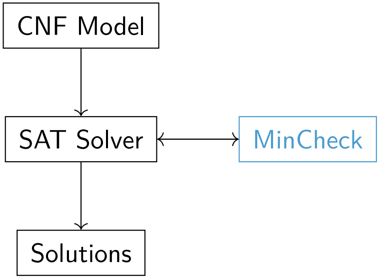
SAT MODULO SYMMETRIES [KS21]

- How:
  1. Obtain a partial interpretation from the SAT solver.
  2. Check whether the assignment can be extended to a complete assignment that is lexicographically minimal.
  3. If not, force the solver to abort the current branch of the search tree by learning a new clause.



SAT MODULO SYMMETRIES [KS21]

- How:
  1. Obtain a partial interpretation from the SAT solver.
  2. Check whether the assignment can be extended to a complete assignment that is lexicographically minimal.
  3. If not, force the solver to abort the current branch of the search tree by learning a new clause.
- This procedure needs to take into account:
  - the (encoding of) the mathematical problem,
    - i.e., the (encoding of) the cycle set definition.
  - and the structure of the set of isomorphisms.
    - i.e., all permutations that fix the diagonal (if one is fixed).



SAT MODULO SYMMETRIES [KS21]

FOR CYCLE SETS

- Given a formula ψ over variables Σ (modelling the cycle set definition),
- With a complete, satisfying assignment α of Σ, we associate a cycle set $\mathbf{C}^{\alpha}$ where for all cells $(i, j) \in X \times X$ it holds that:
  - $\mathbf{C}_{i,j} = k$ iff $v_{i,j,k} \in \alpha$.
- We now want to introduce symmetry breaking constraints during the solving phase.
- But, during the solving phase, the full cycle set might not be known yet.
- Hence, we introduce partial cycle sets.

PARTIAL CYCLE SETS

Partial Cycle Set

A partial cycle set of size n is a matrix $\mathbf{P} \in (2^X)^{n \times n}$ with $X = \{1, \ldots, n\}$, where each cell $c \in X \times X$ of $\mathbf{P}$ represents a non-empty domain $\mathbf{P}_c \subseteq X$ of values that are still possible.

PARTIAL CYCLE SETS

Partial Cycle Set

A partial cycle set of size n is a matrix $\mathbf{P} \in (2^X)^{n \times n}$ with $X = \{1, \ldots, n\}$, where each cell $c \in X \times X$ of $\mathbf{P}$ represents a non-empty domain $\mathbf{P}_c \subseteq X$ of values that are still possible.

- With a partial assignment α of Σ, we associate a partial cycle set $\mathbf{P}^\alpha$ where for all cells $(x, y) \in X \times X$ it holds that:
  - $\mathbf{P}_{x,y} = \{x \in X \mid \neg c_{i,j,x} \notin \alpha\}$.
- In other words, $\mathbf{P}^\alpha$ consist of the values that can still be true according to α.

PARTIAL CYCLE SETS

EXAMPLE

	1	2	3
1	$\{2\}$	$\{1\}$	$\{3\}$
2	$\{2,3\}$	$\{1\}$	$\{2,3\}$
3	$\{1,2\}$	$\{1,2\}$	$\{3\}$

$\mathbf{P} \in (2^X)^{n\times n}$

→

	1	2	3
1	2	1	3
2	2	1	3
3	1	2	3

$\mathbf{C}_1 \in \mathcal{X}(\mathbf{P})$

	1	2	3
1	2	1	3
2	3	1	2
3	2	1	3

$\mathbf{C}_2 \in \mathcal{X}(\mathbf{P})$

$\ldots$

PARTIAL CYCLE SETS

EXAMPLE

	1	2	3
1	$\{2\}$	$\{1\}$	$\{3\}$
2	$\{2,3\}$	$\{1\}$	$\{2,3\}$
3	$\{1,2\}$	$\{1,2\}$	$\{3\}$

$\mathbf{P} \in (2^X)^{n \times n}$

	1	2	3
1	2	1	3
2	2	1	3
3	1	2	3

$\mathbf{C}_1 \in \mathcal{X}(\mathbf{P})$

	1	2	3
1	2	1	3
2	3	1	2
3	2	1	3

$\mathbf{C}_2 \in \mathcal{X}(\mathbf{P})$

...

$\preceq$-minimality

A partial cycle set $\mathbf{P}$ is $\preceq$-minimal if it can be extended to a $\preceq$-minimal cycle set.

PARTIAL CYCLE SETS

LEXICOGRAPHIC MINIMALITY [KS21]

- If for all extended cycle sets $\mathbf{C} \in \mathcal{X}(\mathbf{P})$ there exists an isomorphism π s.t. $\pi(\mathbf{C}) \prec \mathbf{C}$:
  - $\mathbf{P}$ can not be $\preceq$-minimal.
  - But: hard to decide this...

PARTIAL CYCLE SETS

LEXICOGRAPHIC MINIMALITY [KS21]

- If for all extended cycle sets $\mathbf{C} \in \mathcal{X}(\mathbf{P})$ there exists an isomorphism π s.t. $\pi(\mathbf{C}) \prec \mathbf{C}$:
  - $\mathbf{P}$ can not be $\preceq$-minimal.
  - But: hard to decide this...
- If there exists an isomorphism π s.t. $\pi(\mathbf{C}) \prec \mathbf{C}$ for all extended cycle sets $\mathbf{C} \in \mathcal{X}(\mathbf{P})$:
  - $\mathbf{P}$ can not be $\preceq$-minimal.
  - We call π a witness of non-minimality of $\mathbf{P}$!

FINDING WITNESSES

- In order to find these witnesses, we need:
  1. A way to apply permutations to partial cycle sets.
  2. An order $\lhd$ over partial cycle sets,
     - s.t. if $\mathbf{P} \lhd \mathbf{P}'$ then $\mathbf{C} \prec \mathbf{C}'$ for all extensions $\mathbf{C} \in \mathcal{X}(\mathbf{P})$ and $\mathbf{C}' \in \mathcal{X}(\mathbf{P}')$.

FINDING WITNESSES

- In order to find these witnesses, we need:
  1. A way to apply permutations to partial cycle sets.
  2. An order $\lhd$ over partial cycle sets,
     - s.t. if $\mathbf{P} \lhd \mathbf{P}'$ then $\mathbf{C} \prec \mathbf{C}'$ for all extensions $\mathbf{C} \in \mathcal{X}(\mathbf{P})$ and $\mathbf{C}' \in \mathcal{X}(\mathbf{P}')$.

- If we can find a permutation π for which $\pi(\mathbf{P}) \lhd \mathbf{P}$, we have that $\pi(\mathbf{C}) \prec \mathbf{C}$ for all extensions $\mathbf{C} \in \mathcal{X}(\mathbf{P})$.
- In other words, we can decide that π is a witness of non-minimality.

PARTIAL CYCLE SET

APPLYING A PERMUTATION

- Given a partial cycle set $\mathbf{P} \in (2^X)^{n \times n}$ and a permutation $\pi : X \to X$:

$$\pi(\mathbf{P}_{i,j}) = \{\pi^{-1}(x) \mid x \in \mathbf{P}_{\pi(i),\pi(j)}\}.$$

- For example, given $\mathbf{P}$ and $\pi = (12)$:

$$\mathbf{P} = \begin{bmatrix} 1 & 2 & 4 & 3 \\ 1 & 2 & 3 & 4 \\ 1 & \{2,4\} & 3 & \{2,4\} \\ 1 & \{2,3\} & \{2,3\} & 4 \end{bmatrix} \qquad \mathbf{P}_{\pi(i),(j)} = \begin{bmatrix} 2 & 1 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ \{2,4\} & 1 & 3 & \{2,4\} \\ \{2,3\} & 1 & \{2,3\} & 4 \end{bmatrix}$$

PARTIAL CYCLE SET

APPLYING A PERMUTATION

- Given a partial cycle set $\mathbf{P} \in (2^X)^{n \times n}$ and a permutation $\pi : X \to X$:

$$\pi(\mathbf{P}_{i,j}) = \{\pi^{-1}(x) \mid x \in \mathbf{P}_{\pi(i),\pi(j)}\}.$$

- For example, given $\mathbf{P}$ and $\pi = (12)$:

$$\mathbf{P} = \begin{bmatrix} 1 & 2 & 4 & 3 \\ 1 & 2 & 3 & 4 \\ 1 & \{2,4\} & 3 & \{2,4\} \\ 1 & \{2,3\} & \{2,3\} & 4 \end{bmatrix} \qquad \pi(\mathbf{P}) = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 4 & 3 \\ \{1,4\} & 2 & 3 & \{1,4\} \\ \{1,3\} & 2 & \{1,3\} & 4 \end{bmatrix}$$

PARTIAL CYCLE SET

APPLYING A PERMUTATION

- Given a partial cycle set $\mathbf{P} \in (2^X)^{n \times n}$ and a permutation $\pi : X \to X$:

$$\pi(\mathbf{P}_{i,j}) = \{\pi^{-1}(x) \mid x \in \mathbf{P}_{\pi(i),\pi(j)}\}.$$

- For example, given $\mathbf{P}$ and $\pi = (12)$:

$$\mathbf{P} = \begin{bmatrix} 1 & 2 & 4 & 3 \\ 1 & 2 & 3 & 4 \\ 1 & \{2,4\} & 3 & \{2,4\} \\ 1 & \{2,3\} & \{2,3\} & 4 \end{bmatrix} \quad \triangleright ? \quad \pi(\mathbf{P}) = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 4 & 3 \\ \{1,4\} & 2 & 3 & \{1,4\} \\ \{1,3\} & 2 & \{1,3\} & 4 \end{bmatrix}$$

PARTIAL CYCLE SET

APPLYING A PERMUTATION

- Given a partial cycle set $\mathbf{P} \in (2^X)^{n \times n}$ and a permutation $\pi : X \to X$:

$$\pi(\mathbf{P}_{i,j}) = \{\pi^{-1}(x) \mid x \in \mathbf{P}_{\pi(i),\pi(j)}\}.$$

- For example, given $\mathbf{P}$ and $\pi = (12)$:

$$\mathbf{P} = \begin{bmatrix} 1 & \{3,4\} & 2 & \{3,4\} \\ 1 & 2 & 3 & 4 \\ 1 & \{2,4\} & 3 & \{2,4\} \\ 1 & \{2,3\} & \{2,3\} & 4 \end{bmatrix} \qquad \triangleright ? \qquad \pi(\mathbf{P}) = \begin{bmatrix} 1 & 2 & 3 & 4 \\ \{3,4\} & 2 & 1 & \{3,4\} \\ \{1,4\} & 2 & 3 & \{1,4\} \\ \{1,3\} & 2 & \{1,3\} & 4 \end{bmatrix}$$

PARTIAL CYCLE SET

ORDERING PARTIAL CYCLE SETS

$\mathbf{P} \lhd \mathbf{P}'$

Given two partial cycle sets $\mathbf{P}$ and $\mathbf{P}'$ we say that $\mathbf{P} \lhd \mathbf{P}'$ iff:

- there is a cell c s.t. $\max \mathbf{P}_c < \min \mathbf{P}'_c$ and
- for all cells $c' < c$: $\max \mathbf{P}_{c'} \leq \min \mathbf{P}'_{c'}$.

PARTIAL CYCLE SET

ORDERING PARTIAL CYCLE SETS

$\mathbf{P} \lhd \mathbf{P}'$

Given two partial cycle sets $\mathbf{P}$ and $\mathbf{P}'$ we say that $\mathbf{P} \lhd \mathbf{P}'$ iff:

- there is a cell c s.t. $\max \mathbf{P}_c < \min \mathbf{P}'_c$ and
- for all cells $c' < c$: $\max \mathbf{P}_{c'} \leq \min \mathbf{P}'_{c'}$.

$\mathbf{P} \unlhd \mathbf{P}'$

Given two partial cycle sets $\mathbf{P}$ and $\mathbf{P}'$ we say that $\mathbf{P} \unlhd \mathbf{P}'$ iff:

- either $\mathbf{P} \lhd \mathbf{P}'$, or for all cells c: $\max \mathbf{P}_c \leq \min \mathbf{P}'_c$.

MINIMALITY CHECK

OVERVIEW

- Goal: decide whether $\exists \pi \in \langle \Pi \rangle$, such that $\pi(\mathbf{P}) \lhd \mathbf{P}$, given
  - a matrix $\mathbf{P}$ representing a partial cycle set, and
  - where the group $\langle \Pi \rangle$ represents the isomorphisms of the problem.
- Considering each π one-by-one is not a feasible option...

MINIMALITY CHECK

BACKTRACKING APPROACH [KS21]

- Represent all possible isomorphism of the problem.
  - i.e. $\pi(1) = [1, 2, 3, 4], \pi(2) = [1, 2, 3, 4], \ldots$
- Make decision
  - i.e. $\pi(1) = [2]$
- Propagate:
  - i.e. $\pi(2) = [1, 3, 4]$
    - Ensure that the partial permutation π can be extended to an isomorphism of the problem.
    - Given the partial cycle set $\mathbf{P}$, ensure that $\pi(\mathbf{P}) \lhd \mathbf{P}$.
- Repeat until:
  - A witness is found.
  - All possibilities have failed.

MINIMALITY CHECK

BACKTRACKING APPROACH [KS21]

- Represent all possible isomorphism of the problem.
  - i.e. $\pi(1) = [1, 2, 3, 4], \pi(2) = [1, 2, 3, 4], \ldots$
- Make decision
  - i.e. $\pi(1) = [2]$
- Propagate:
  - i.e. $\pi(2) = [1, 3, 4]$
    - Ensure that the partial permutation π can be extended to an isomorphism of the problem.
    - Given the partial cycle set $\mathbf{P}$, ensure that $\pi(\mathbf{P}) \lhd \mathbf{P}$.
- Repeat until:
  - A witness is found.
  - All possibilities have failed.

- Issue!
- Sometimes there is no information to propagate
- Worst case complexity of $n!$...

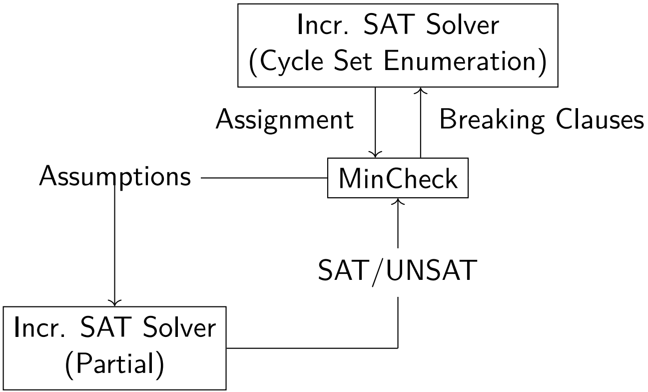
MINIMALITY CHECK

INCREMENTAL, SAT-BASED APPROACH

- Minimality check = combinatorial search problem
  - i.e. given the current (partial) cycle set, does there exist a witness?
- We chose to:
  - Express the problem in CNF.
  - Use an incremental SAT-solver to verify whether the CNF is satisfiable given the current assumptions.
  - If so, we have found a witness of non-minimality for the current cycle set!

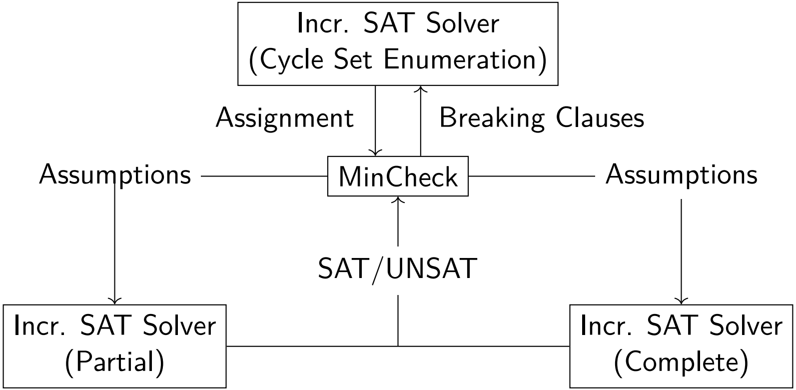
INCREMENTAL, SAT-BASED MINIMALITY CHECK

OVERVIEW



INCREMENTAL, SAT-BASED MINIMALITY CHECK

OVERVIEW



MINIMALITY CHECK

CONSTRUCTING A CLAUSE

- π is a witness of non-minimality!
  - There exists cell $c = (i, j)$ such that:
    - for all cells $c' < c$: $\pi(\mathbf{P})_{c'} \trianglelefteq \mathbf{P}_{c'}$ and,
    - $\pi(\mathbf{P})_c \triangleleft \mathbf{P}_c$.
- So: how do we exclude the current solution (and its extensions?)

MINIMALITY CHECK

CONSTRUCTING A CLAUSE

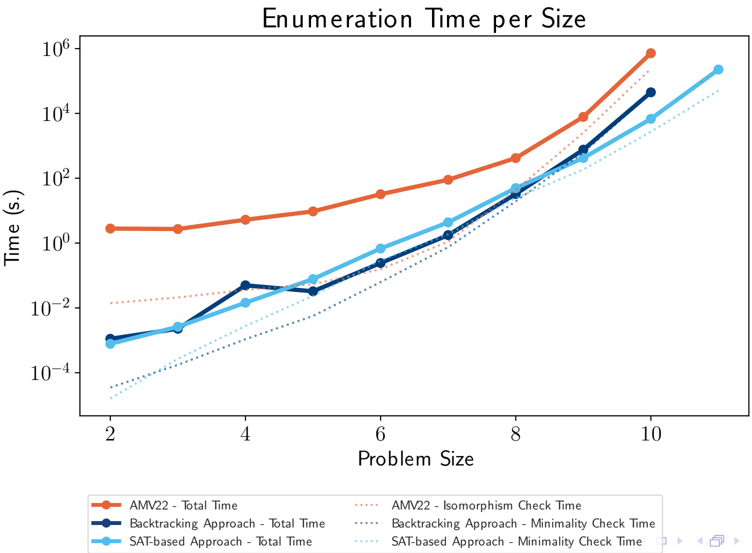
- π is a witness of non-minimality!
  - There exists cell $c = (i, j)$ such that:
    - for all cells $c' < c$: $\pi(\mathbf{P})_{c'} \trianglelefteq \mathbf{P}_{c'}$ and,
    - $\pi(\mathbf{P})_c \triangleleft \mathbf{P}_c$.
- So: how do we exclude the current solution (and its extensions?)

- We add a clause expressing that (at least) one of these conditions is different:
  - $\max \pi(\mathbf{P})_c$ becomes larger than or equal to $\min \mathbf{P}_c$,
  - or for at least one of the cells $c' < c$; $\max \pi(\mathbf{P})_{c'}$ becomes strictly larger than $\min \mathbf{P}_{c'}$,
  - or the solver needs to backtrack.

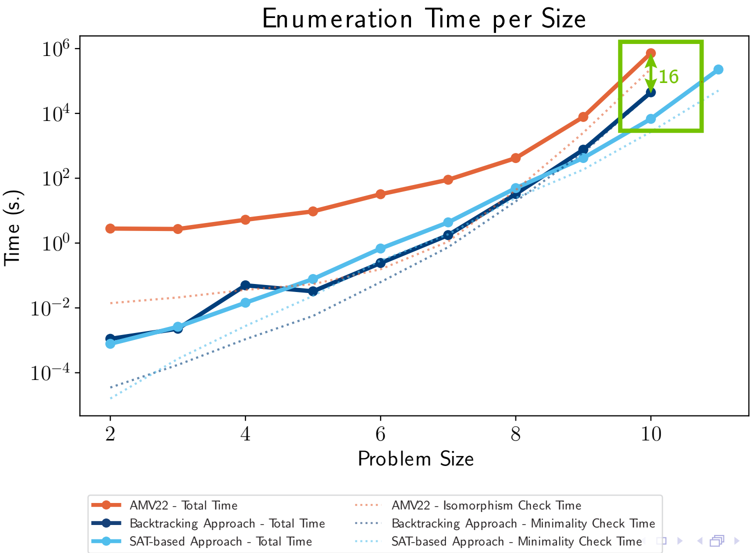
IMPLEMENTATION

- We use CaDiCaL [BFFH20] with the IPASIR-UP API [FNP+23];
  - to keep track of the current assignment,
  - to add clauses if a useful permutation is found,
  - and to find witnesses.
- The implementation and database are available on GitLab.
- Experiments were performed on a machine with
  - an AMD(R) Genoa-X CPU,
  - running Rocky Linux 8.9,
  - with Linux kernel 4.18.0.

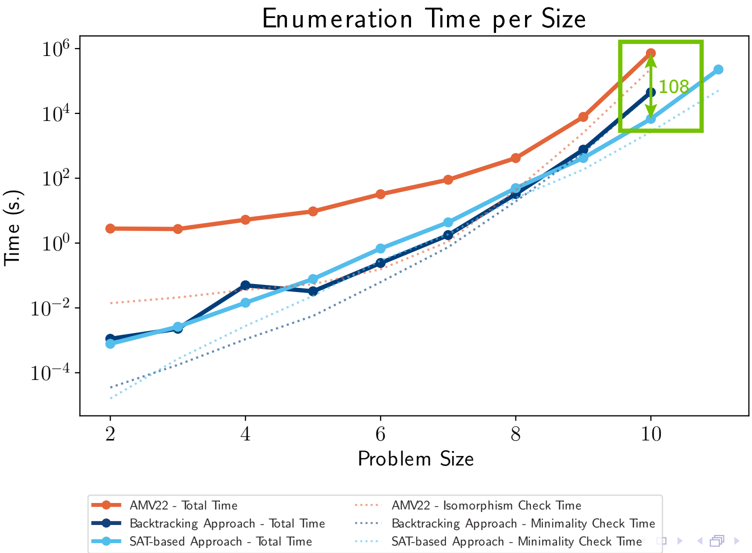
COMPARING RESULTS



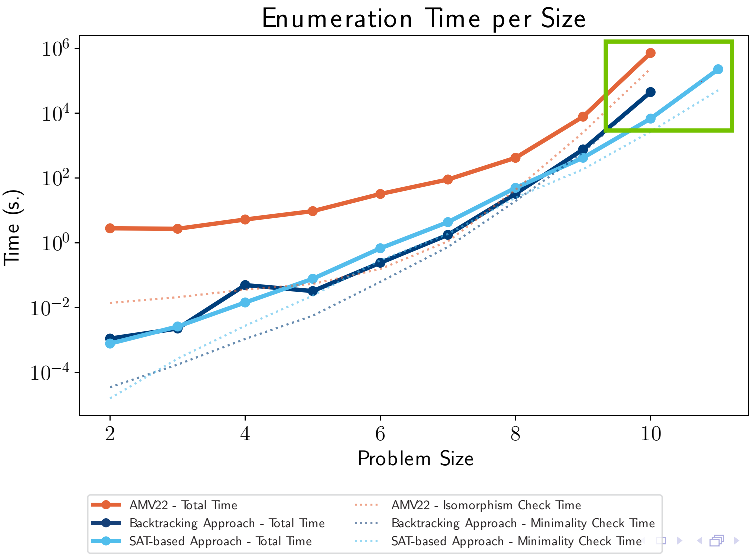
COMPARING RESULTS



COMPARING RESULTS



COMPARING RESULTS



FUTURE WORK

- Refining incremental approach

FUTURE WORK

- Refining incremental approach
- Certifying the results
  - We obtain the same results as [AMV22], but that only means that we are either both correct or both wrong.
  - However, how do we verify this?

FUTURE WORK

- Refining incremental approach
- Certifying the results
  - We obtain the same results as [AMV22], but that only means that we are either both correct or both wrong.
  - However, how do we verify this?
- Enumerating related structures
  - Racks,
    - used to enumerate skew cycle sets.
  - Skew Cycle Sets,
    - correspond to non-degenerate set-theoretic solutions.
  - Biquandles,
    - applications in knot theory.

FUTURE WORK

- Refining incremental approach
- Certifying the results
  - We obtain the same results as [AMV22], but that only means that we are either both correct or both wrong.
  - However, how do we verify this?

- Enumerating related structures
  - Racks,
    - used to enumerate skew cycle sets.
  - Skew Cycle Sets,
    - correspond to non-degenerate set-theoretic solutions.
  - Biquandles,
    - applications in knot theory.
- Generelizing the approach?

REFERENCES

[AMV22] Özgür Akgün, M. Mereb, and Leandro Vendramin. Enumeration of set-theoretic solutions to the yang–baxter equation. *Mathematics of Computation*, jan 2022.

[Bax72] Rodney J. Baxter. Partition function of the eight-vertex lattice model. *Annals of Physics*, 70(1):193–228, 1972.

[BFFH20] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In Tomas Balyo, Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.

[BGMN22] Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Certified symmetry and dominance breaking for combinatorial optimisation. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 3698–3707. AAAI Press, 2022.

[Dri92] V. G. Drinfeld. *On some unsolved problems in quantum group theory*, page 1–8. Springer Berlin Heidelberg, 1992.

REFERENCES

[FNP+23] Katalin Fazekas, Aina Niemetz, Mathias Preiner, Markus Kirchweger, Stefan Szeider, and Armin Biere. IPASIR-UP: user propagators for CDCL. In Meena Mahajan and Friedrich Slivovsky, editors, *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy*, volume 271 of *LIPIcs*, pages 8:1–8:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[KS21] Markus Kirchweger and Stefan Szeider. SAT modulo symmetries for graph generation. In Laurent D. Michel, editor, *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, volume 210 of *LIPIcs*, pages 34:1–34:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[KSS22] Markus Kirchweger, Manfred Scheucher, and Stefan Szeider. A SAT attack on rota's basis conjecture. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, volume 236 of *LIPIcs*, pages 4:1–4:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Yan67] Chen Ning Yang. Some exact results for the many-body problem in one dimension with repulsive delta-function interaction. *Phys. Rev. Lett.*, 19:1312–1315, Dec 1967.

YANG-BAXTER EQUATION

DEFINITION

Yang-Baxter Equation [Yan67, Bax72]

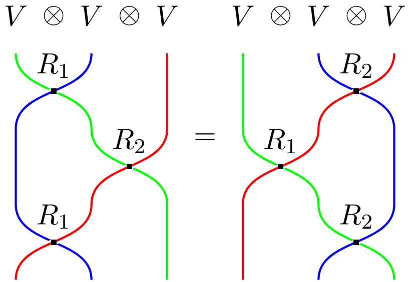
A solution to the Yang-Baxter equation (YBE) is a pair (V, R), where V is a vector space and $R : V \otimes V \to V \otimes V$ is a map such that in $(V \otimes V \otimes V)$,

$$R_1 R_2 R_1 = R_2 R_1 R_2,$$

where R_i acts as R on components i and $i+1$, and as the identity on the other component.

YANG-BAXTER EQUATION

DEFINITION



$$R_1 R_2 R_1 = R_2 R_1 R_2$$

Figure: A visual representation of the Yang-Baxter equation.

YANG-BAXTER EQUATION

DEFINITION

Set-Theoretic Yang-Baxter Equation (YBE) [Dri92]

A set-theoretic solution to the YBE is a pair (X, r), where X is a non-empty set and $r : X^2 \to X^2$ is a map such that in X^3,

$$r_1 r_2 r_1 = r_2 r_1 r_2, \qquad \text{(the Yang-Baxter Equation)}$$

where r_i acts as r on components i and $i + 1$ and as the identity on the other component.

- These solutions are a subset of the solutions to the *original* Yang-Baxter equation.
- A set-theoretic solution is called involutive if $r^2 = id_{X \times X}$.
- A set-theoretic solution with $r(x, y) = (\sigma_x(y), \tau_y(x))$is called non-degenerate if the maps σ_x and τ_x are bijective for all $x \in X$.

CYCLE SETS

- A cycle set $(X, \cdot)$ consists of a non-empty set X and a binary operation $\cdot$ on X s.t.:
  1. for all $x \in X$, the map $\phi_x : X \to X : y \mapsto x \cdot y$ is bijective,
  2. for all $x, y, z \in X$, $(x \cdot y) \cdot (x \cdot z) = (y \cdot x) \cdot (y \cdot z)$,
  3. the map $X \to X : x \mapsto x \cdot x$ is bijective. (non-degenerate)

CYCLE SETS

- A cycle set $(X, \cdot)$ consists of a non-empty set X and a binary operation $\cdot$ on X s.t.:
  1. for all $x \in X$, the map $\phi_x : X \to X : y \mapsto x \cdot y$ is bijective,
  2. for all $x, y, z \in X$, $(x \cdot y) \cdot (x \cdot z) = (y \cdot x) \cdot (y \cdot z)$,
  3. the map $X \to X : x \mapsto x \cdot x$ is bijective. (non-degenerate)

- Each finite cycle set $(X, \cdot)$ can also be represented by a matrix $\mathbf{C}$ where
  - $\mathbf{C} \in X^{|X| \times |X|}$, and
  - $\mathbf{C}_{x,y} = x \cdot y$ for all $x, y \in X$.

CYCLE SETS

- A cycle set $(X, \cdot)$ consists of a non-empty set X and a binary operation $\cdot$ on X s.t.:
  1. for all $x \in X$, the map $\phi_x : X \to X : y \mapsto x \cdot y$ is bijective,
  2. for all $x, y, z \in X$, $(x \cdot y) \cdot (x \cdot z) = (y \cdot x) \cdot (y \cdot z)$,
  3. the map $X \to X : x \mapsto x \cdot x$ is bijective. (non-degenerate)

- Each finite cycle set $(X, \cdot)$ can also be represented by a matrix $\mathbf{C}$ where
  1. for all $x \in X$, $\mathbf{C}_{x,y} \neq \mathbf{C}_{x,z}$ for all $y, z \in X$ with $y \neq z$,
  2. for all $x, y, z \in X$, $\mathbf{C}_{\mathbf{C}_{x,y}, \mathbf{C}_{x,z}} = \mathbf{C}_{\mathbf{C}_{y,x}, \mathbf{C}_{y,z}}$,
  3. for all $x \in X$, $\mathbf{C}_{x,x} \neq \mathbf{C}_{y,y}$ for all $y \in X$ with $y \neq x$. (non-degenerate)

CYCLE SETS

- A cycle set $(X, \cdot)$ consists of a non-empty set X and a binary operation $\cdot$ on X s.t.:
  1. for all $x \in X$, the map $\phi_x : X \to X : y \mapsto x \cdot y$ is bijective,
  2. for all $x, y, z \in X$, $(x \cdot y) \cdot (x \cdot z) = (y \cdot x) \cdot (y \cdot z)$,
  3. the map $X \to X : x \mapsto x \cdot x$ is bijective. (non-degenerate)
- Each finite cycle set $(X, \cdot)$ can also be represented by a matrix $\mathbf{C}$ where
  1. for all $x \in X$, $\mathbf{C}_{x,y} \neq \mathbf{C}_{x,z}$ for all $y, z \in X$ with $y \neq z$,
  2. for all $x, y, z \in X$, $\mathbf{C}_{\mathbf{C}_{x,y},\mathbf{C}_{x,z}} = \mathbf{C}_{\mathbf{C}_{y,x},\mathbf{C}_{y,z}}$,
  3. for all $x \in X$, $\mathbf{C}_{x,x} \neq \mathbf{C}_{y,y}$ for all $y \in X$ with $y \neq x$. (non-degenerate)
- Two cycle sets $(X, \cdot)$ and $(X, \times)$ are called isomorphic when there exists a bijection $f : X \to X$ such that $f(x \cdot y) = f(x) \times f(y)$.

CNF MODEL CYCLE SETS

- for each $i, j, x \in X$, the Boolean variable $v_{i,j,x}$ is true iff $\mathbf{C}_{i,j} = x$
- Ensure that each matrix entry is assigned exactly one value;
  - for each $i, j \in X$: $\texttt{exactlyOne}([v_{i,j,k} \mid k \in X])$

CNF MODEL CYCLE SETS

- for each $i, j, x \in X$, the Boolean variable $v_{i,j,x}$ is true iff $\mathbf{C}_{i,j} = x$
- Ensure that each matrix entry is assigned exactly one value;
  - for each $i, j \in X$: $\texttt{exactlyOne}([v_{i,j,k} \mid k \in X])$
- Rows contain unique values;
  - for each $i, k \in X$: $\texttt{exactlyOne}([v_{i,j,k} \mid j \in X])$

CNF MODEL CYCLE SETS

- for each $i, j, x \in X$, the Boolean variable $v_{i,j,x}$ is true iff $\mathbf{C}_{i,j} = x$
- Ensure that each matrix entry is assigned exactly one value;
  - for each $i, j \in X$: $\texttt{exactlyOne}([v_{i,j,k} \mid k \in X])$
- Rows contain unique values;
  - for each $i, k \in X$: $\texttt{exactlyOne}([v_{i,j,k} \mid j \in X])$
- The diagonal contains unique values;
  - $\texttt{exactlyOne}([v_{i,i,k} \mid i \in X])$

CNF MODEL CYCLE SETS

- for each $i, j, x \in X$, the Boolean variable $v_{i,j,x}$ is true iff $\mathbf{C}_{i,j} = x$
- Ensure that each matrix entry is assigned exactly one value;
  - for each $i, j \in X$: $\texttt{exactlyOne}([v_{i,j,k} \mid k \in X])$
- Rows contain unique values;
  - for each $i, k \in X$: $\texttt{exactlyOne}([v_{i,j,k} \mid j \in X])$
- The diagonal contains unique values;
  - $\texttt{exactlyOne}([v_{i,i,k} \mid i \in X])$

- for all $i, j, k, b \in X$ with $i < j$, the Boolean variable $y_{i,j,k,b}$ is true iff $\mathbf{C}_{\mathbf{C}_{i,j},\mathbf{C}_{i,k}} = \mathbf{C}_{\mathbf{C}_{k,i},\mathbf{C}_{k,j}} = b$

CNF MODEL CYCLE SETS

- for each $i, j, x \in X$, the Boolean variable $v_{i,j,x}$ is true iff $\mathbf{C}_{i,j} = x$
- Ensure that each matrix entry is assigned exactly one value;
  - for each $i, j \in X$: $\texttt{exactlyOne}([v_{i,j,k} \mid k \in X])$
- Rows contain unique values;
  - for each $i, k \in X$: $\texttt{exactlyOne}([v_{i,j,k} \mid j \in X])$
- The diagonal contains unique values;
  - $\texttt{exactlyOne}([v_{i,i,k} \mid i \in X])$

- for all $i, j, k, b \in X$ with $i < j$, the Boolean variable $y_{i,j,k,b}$ is true iff $\mathbf{C}_{\mathbf{C}_{i,j},\mathbf{C}_{i,k}} = \mathbf{C}_{\mathbf{C}_{k,i},\mathbf{C}_{k,j}} = b$
  - for all $i, j, k, x, y, b \in X$ where $i < j$:
    - $\neg v_{i,j,x} \vee \neg v_{i,k,y} \vee \neg v_{x,y,b} \vee y_{i,j,k,b}$
    - $\neg v_{j,i,x} \vee \neg v_{j,k,y} \vee \neg v_{x,y,b} \vee y_{i,j,k,b}$

CNF MODEL CYCLE SETS

- for each $i, j, x \in X$, the Boolean variable $v_{i,j,x}$ is true iff $\mathbf{C}_{i,j} = x$
- Ensure that each matrix entry is assigned exactly one value;
  - for each $i, j \in X$: $\texttt{exactlyOne}([v_{i,j,k} \mid k \in X])$
- Rows contain unique values;
  - for each $i, k \in X$: $\texttt{exactlyOne}([v_{i,j,k} \mid j \in X])$
- The diagonal contains unique values;
  - $\texttt{exactlyOne}([v_{i,i,k} \mid i \in X])$

- for all $i, j, k, b \in X$ with $i < j$, the Boolean variable $y_{i,j,k,b}$ is true iff $\mathbf{C}_{\mathbf{C}_{i,j},\mathbf{C}_{i,k}} = \mathbf{C}_{\mathbf{C}_{k,i},\mathbf{C}_{k,j}} = b$
  - for all $i, j, k, x, y, b \in X$ where $i < j$:
    - $\neg v_{i,j,x} \vee \neg v_{i,k,y} \vee \neg v_{x,y,b} \vee y_{i,j,k,b}$
    - $\neg v_{j,i,x} \vee \neg v_{j,k,y} \vee \neg v_{x,y,b} \vee y_{i,j,k,b}$
  - for all $i, j, k \in X$ where $i < j$:
    - $\texttt{exactlyOne}([y_{i,j,k,b} \mid b \in X])$

TIME ANALYSIS

		# Sols.	Backtracking Approach Time (% of total time)	Backtracking Approach Solver % of Time	Backtracking Approach MinCheck % of Time	Incr. SAT Approach Time (% of total time)	Incr. SAT Approach Solver % of Time	Incr. SAT Approach MinCheck % of Time
8	id	2041	15.45s. (47.33)	10.02	89.98	4.53s. (29.27)	41.21	58.79
	(12)	4988	2.87s. (8.78)	41.76	58.24	4.08s. (8.21)	42.53	57.47
	(12)(34)	7030	2.48s. (7.59)	59.17	40.83	4.52s. (9.10)	45.62	54.38
9	id	15534	514.63s. (67.67)	2.85	92.41	135.86s. (32.25)	41.77	58.23
	(12)	41732	68.82s. (9.05)	18.07	75.91	37.68s. (8.95)	47.49	52.50
	(12)(34)	61438	37.69s. (4.96)	42.55	46.78	41.99s. (9.97)	51.72	48.28
10	id	150957	35 396.79s. (79.02)	0.55	98.03	1 073.65s. (15.80)	36.30	63.70
	(12)	474153	3 998.35s. (8.93)	6.12	92.08	605.32s. (8.91)	52.35	47.65
	(12)(34)	807084	1 380.82s. (3.08)	30.94	63.98	817.65s. (12.03)	59.14	40.85

FUTURE WORK

CERTIFYING THE RESULS

- How do we know whether these results are correct?
  - We obtain the same results as [AMV22], but that only means that we are either both correct or both wrong.
- Many SAT Solvers are verifiable
  - They produce a solution and a machine-verifiable proof for this solution
  - This proof is then verified together with the CNF formula
- This is also the case for `CaDiCaL`, even with the `SMS` framework [KSS22]
  - However: only verified if each clause is added with a good reason
- So, how do we know whether the added breaking clauses were correct?
  - `VeriPB` can verify static symmetry breaking [BGMN22]
  - `CaDiCaL` comes with `VeriPB`
- How do we verify whether we have enumerated exactly one solution per isomorphism class?
  - Non-trivial, we need information about the problem. . .
  - The symmetries of the CNF might not be equivalent to the isomorphisms of the problem...