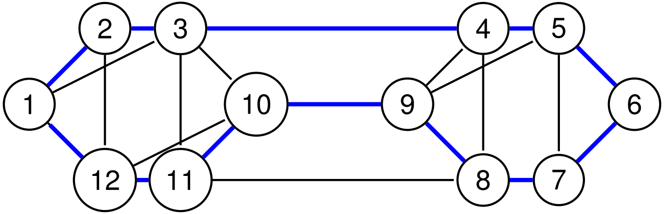
SAT-based CEGAR Method for the Hamiltonian Cycle Problem Enhanced by Cut-Set Constraints

Ryoga Ohashi[1], Takehide Soh[1], Daniel Le Berre[2],
Hidetomo Nabeshima[3], Mutsunori Banbara[4], Katsumi Inoue[5],
Naoyuki Tamura[1]

[1]Kobe University,
[2]Univ. Artois, CNRS, UMR 8188 CRIL,
[3]Yamanashi University,
[4]Nagoya University,
[5]National Institute of Informatics

SAT2025
2025.8.15

Hamiltonian Cycle Problem (HCP)



- HCP is the problem of determining whether a given graph has a Hamiltonian cycle, that is, a cycle that visits every vertex exactly once.
- HCP is one of the 21 NP-complete problems identified by Karp in 1972 [Karp, 1972].
- Active research has been conducted on fast solution methods, including the international competition on the Hamiltonian Cycle Problem, the "Flinders Hamiltonian Cycle Project (FHCP) Challenge".

Existing Approaches: TSP-based Methods

HCP is a special case of the TSP, and can often be solved efficiently using TSP solvers.

- Concorde [Applegate+, 2006]
- LKH [Helsgaun, 2000]
- Snakes and Ladders Heuristic [Baniasadi+, 2019]

- However, there exist instances that cannot be solved by TSP solvers.
- The instances in the FHCP Challenge are specifically designed to be such cases and cannot be efficiently solved using TSP-based methods.

Alternative Approaches: SAT-based Methods

- The development of fast SAT solvers has led to the proposal of many SAT-based approaches for HCP.

Existing SAT-based Methods for HCP

- Absolute/Relative Encoding [Prestwich, 2003]
- Permutations-based Encoding [Velev+, 2009]
- SAT-based CEGAR for HCP [Soh+, 2014]
- Binary Adder Encoding [Zhou, 2020]
- CRT Encoding [Heule, 2021]

- Recent approaches show good performance on some FHCP Challenge instances.
- However, there are still unsolvable instances.

This talk

Goal

To develop an efficient SAT-based method for HCP.

Idea

Accelerate SAT-based CEGAR for HCP by using cut-set [Dantzig+, 1954].

Outline

- Constraints model for HCP
- Existing SAT-based CEGAR approach (conflict constraints)
- Proposed SAT-based CEGAR approach with cutset (support) constraints
- Experimental evaluation

Constraints model for HCP

Given a graph $G = (V, E)$,
if there exists a subgraph $G' = (V, E')$ of G that satisfies the following constraints, then G is a Hamiltonian graph.

(1) Degree Constraint

- For every vertex $v \in V$, the degree of v must be exactly 2.

(2) Connectivity Constraint

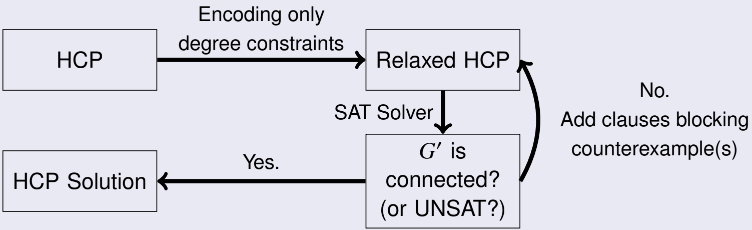
- G' is connected.

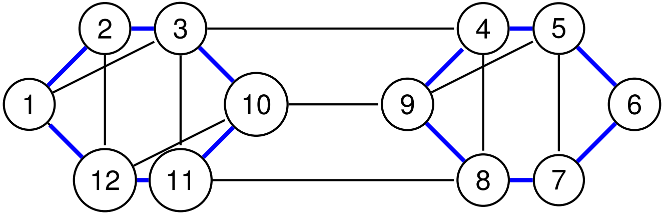
- Degree Constraint = Cardinality Constraint (Easy)
- Connectivity Constraint = ??? (Problematic)

Previous studies have focused on how to encode Connectivity Constraint.

SAT-based CEGAR for HCP [Soh+, 2014]

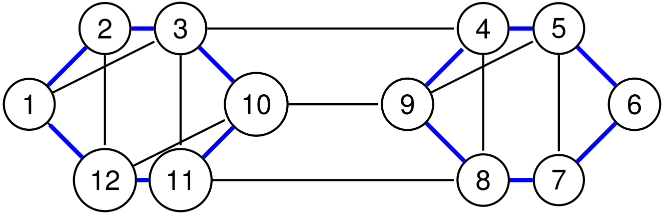
Procedure





Counterexample (G' is not connected)

How to block counterexample(s)?



Counterexample (G' is not connected)

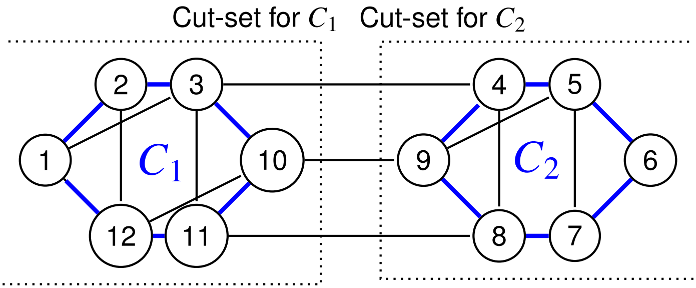
(Naive) negate model found: prune one counterexample.

$$\neg x_{1,2} \vee \neg x_{2,3} \vee \neg x_{3,10} \vee \neg x_{10,11} \vee \neg x_{11,12} \vee \neg x_{12,1} \vee \neg x_{4,5} \vee \neg x_{5,6} \vee \neg x_{6,7} \vee$$
$$\neg x_{7,8} \vee \neg x_{8,9} \vee \neg x_{9,4}$$

[Soh+, 2014] negate each subcycle: prune counterexamples including subcycle found.

$$(\neg x_{1,2} \vee \neg x_{2,3} \vee \neg x_{3,10} \vee \neg x_{10,11} \vee \neg x_{11,12} \vee \neg x_{12,1}) \wedge$$
$$(\neg x_{4,5} \vee \neg x_{5,6} \vee \neg x_{6,7} \vee \neg x_{7,8} \vee \neg x_{8,9} \vee \neg x_{9,4})$$

How to block counterexample(s)?



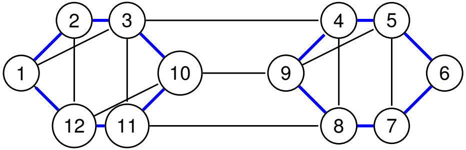
Counterexample (G' is not connected)

(Proposal) activate one of cut-set edges (support clauses).

$$x_{3,4} \vee x_{10,9} \vee x_{11,8}$$

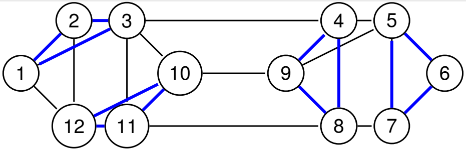
In this example, the cut-set for C_1 and C_2 is identical, so only one clause is needed, but normally one clause is required for each subcycle.

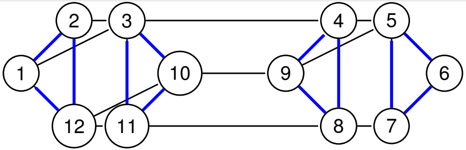
Blocking a "class" of cycles



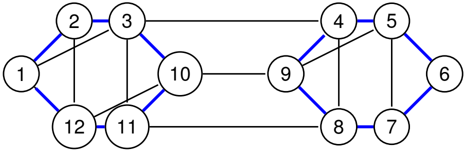
$$\Psi_{sub} = (\neg x_{1,2} \vee \neg x_{2,3} \vee \neg x_{3,10} \vee \neg x_{10,11} \vee \neg x_{11,12} \vee \neg x_{12,1}) \wedge$$
$$(\neg x_{4,5} \vee \neg x_{5,6} \vee \neg x_{6,7} \vee \neg x_{7,8} \vee \neg x_{8,9} \vee \neg x_{9,4})$$
$$\Psi_{cut} = x_{3,4} \vee x_{10,9} \vee x_{11,8}$$

Ψ_{sub} cannot block the following counterexamples, whereas Ψ_{cut} can.
Ψ_{cut} can always prune at least as many counterexamples as Ψ_{sub} (See Proposition 2 in the paper).





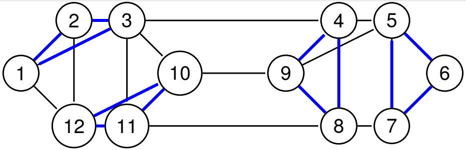
Blocking a "class" of cycles

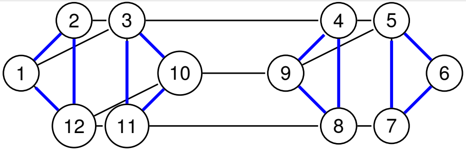


$$\Psi_{sub} = (\neg x_{1,2} \vee \neg x_{2,3} \vee \neg x_{3,10} \vee \neg x_{10,11} \vee \neg x_{11,12} \vee \neg x_{12,1}) \wedge$$
$$(\neg x_{4,5} \vee \neg x_{5,6} \vee \neg x_{6,7} \vee \neg x_{7,8} \vee \neg x_{8,9} \vee \neg x_{9,4})$$
$$\Psi_{cut} = x_{3,4} \vee x_{10,9} \vee x_{11,8}$$

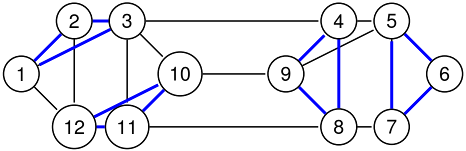
Main characteristic

The cut-set clause $x_{3,4} \vee x_{10,9} \vee x_{11,8}$ blocks not just the subcycle itself, but also the subcycles forming a closed subgraph induced by that subcycle.





Anti-patterns and Merging: Abstraction



For the above counterexample, Ψ_{cut} is as follows.

$$\begin{aligned}\Psi_{cut} = &(x_{3,4} \vee x_{3,10} \vee x_{3,11} \vee x_{2,12} \vee x_{1,12}) \wedge \\ &(x_{1,12} \vee x_{2,12} \vee x_{3,11} \vee x_{3,10} \vee x_{9,10} \vee x_{8,11}) \wedge \\ &(x_{3,4} \vee x_{9,10} \vee x_{8,11} \vee x_{4,5} \vee x_{7,8}) \wedge \\ &(x_{4,5} \vee x_{5,9} \vee x_{7,8})\end{aligned}$$

Ψ_{cut} prunes the same number of counterexamples as Ψ_{sub} in this case since all subcycles are not decomposed into smaller subcyles.

- In the proposed method, we also enhance the pruning capability by merging subcycles.

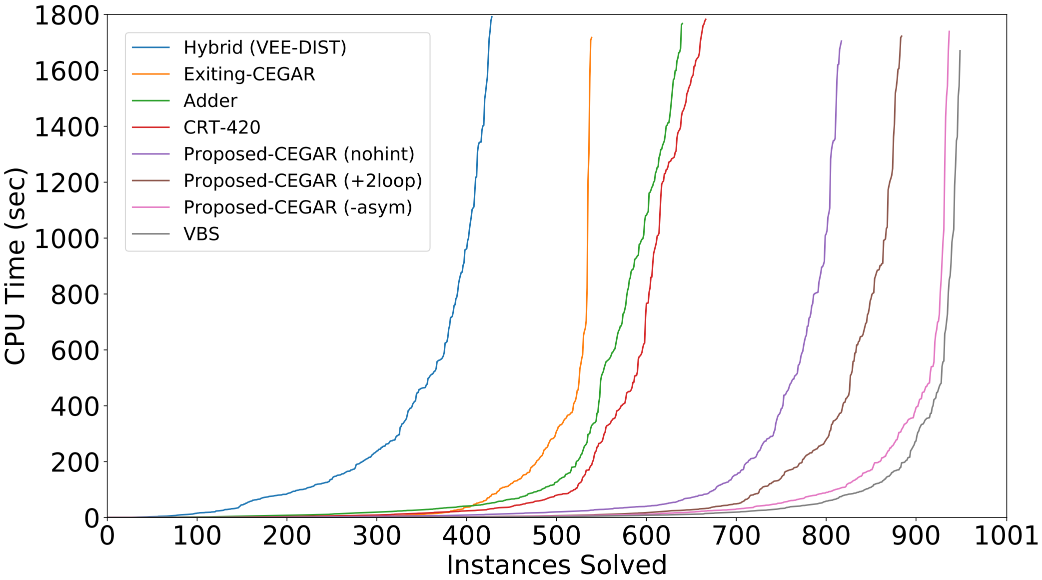
Experimental Environment and Benchmarks

- We implemented both the existing and proposed SAT-based CEGAR solvers in Rust. Other solvers were evaluated using publicly available source code.
- Optimizations in the Proposed SAT-based CEGAR:
  - 2loop: Prohibits 2-vertex loops [Soh+, 2014].
  - asym: Symmetry breaking [Heule, 2021].
  - merge: Cycle merging strategy.
- Encoding for Degree Constraints: Seq. Counter [Sinz, 2005]
- Benchmark:
  - All HCP instances (1,001 in total) from the Flinders Hamiltonian Cycle Project (FHCP) [Haythorpe, 2019]
  - All instances are SAT (i.e., Hamiltonian cycles exist).
- CPU: 2.5GHz, Memory: 64GB
- SAT Solver: CaDiCaL version 1.9.4
- Time Limit: 30 minutes per instance

Ablation Study on Optimization Techniques

|V|	#inst.	nohint	+2loop	+asym	+merge	-merge	-asym	-2loop	all	VBS
~ 1000	171	171	171	171	170	171	171	171	171	171
~ 2000	165	165	165	162	165	165	165	163	164	165
~ 3000	177	173	177	172	175	174	175	173	175	177
~ 4000	185	152	166	151	158	166	166	159	167	169
~ 5000	128	86	101	90	96	107	118	98	118	119
~ 6000	80	35	52	37	47	55	76	51	75	77
~ 7000	55	20	35	23	27	37	44	29	44	47
~ 8000	28	9	11	9	13	13	15	12	15	16
~ 9000	10	4	4	5	5	5	5	5	5	5
~ 10000	2	2	2	2	2	2	2	2	2	2
Total	1001	817	884	822	858	895	937	863	936	948

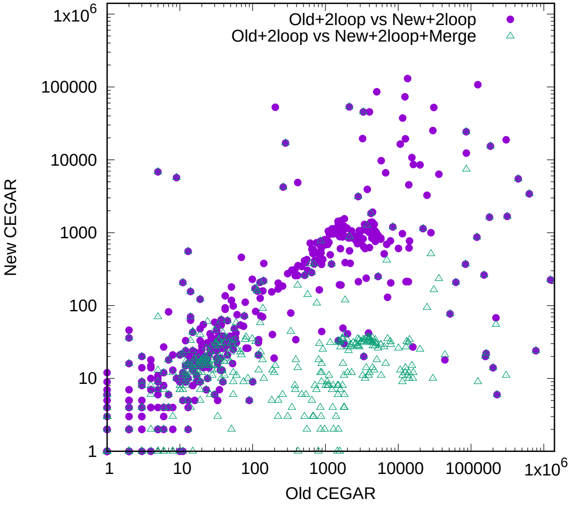
Comparison with Previous SAT-based Methods



Experimental Results: Comparison of SAT-based Methods

Number of Vertices	#Inst.	Adder	CRT-420	CEGAR conflict	CEGAR support
~ 1000	171	170	170	150	171
~ 2000	165	154	152	106	165
~ 3000	177	128	132	89	175
~ 4000	185	94	87	75	166
~ 5000	128	39	54	45	118
~ 6000	80	23	31	36	76
~ 7000	55	18	25	24	44
~ 8000	28	9	10	9	15
~ 9000	10	4	4	4	5
~ 10000	2	1	1	1	2
Total	1001	640	666	539	937

Comparison on CEGAR Iterations



Conclusion and Future Work

- We proposed a modification to the SAT-based CEGAR approach for efficiently solving the Hamiltonian Cycle Problem, by changing how subcycles are prevented.
- Experimental results on benchmark instances from FHCP Challenge showed that the proposed method scales better than previous SAT approaches.
- The proposed method solves 937 out of 1001 instances from FHCP benchmarks: the winner of the FHCP challenge solved 985.

Future Work:

- solve remaining 64 unsolved benchmarks from FHCP
  - Incorporate IPASIR-UP so that we can prune counterexamples every time a subcycle is detected during SAT solving.
  - Incorporate parallel computation.
- Conduct experiments on other HCP benchmarks.

Reference I

[Applegate+, 2006] Applegate, David L., Bixby, Robert E., Chvatál, Vašek , and Cook, William J. (2006).
The Traveling Salesman Problem: A Computational Study.
Princeton University Press.

[Baniasadi+, 2019] Baniasadi, Pouya, Ejov, Vladimir, Filar, Jerzy A, Haythorpe, Michael , and Rossomakhine, Serguei (2019).
Deterministic "snakes and ladders" heuristic for the hamiltonian cycle problem.

[Dantzig+, 1954] Dantzig, George B., Fulkerson, Delbert R. , and Johnson, Selmer M. (1954).
Solution of a large-scale traveling-salesman problem.
Journal of the Operations Research Society of America, 2(4):393–410.

[Haythorpe, 2019] Haythorpe, Michael (2019).
Fhcp challenge set: The first set of structurally difficult instances of the hamiltonian cycle problem.

[Helsgaun, 2000] Helsgaun, Keld (2000).
An effective implementation of the lin–kernighan traveling salesman heuristic.
European Journal of Operational Research, 126(1):106–130.

[Heule, 2021] Heule, Marijn J. H. (2021).
Chinese remainder encoding for hamiltonian cycles.
In Li, Chu-Min and Manyà, Felip, editors, *Theory and Applications of Satisfiability Testing - SAT 2021 - 24th International Conference, Barcelona, Spain, July 5-9, 2021, Proceedings*, volume 12831 of *Lecture Notes in Computer Science*, pages 216–224. Springer.

Reference II

[Karp, 1972] Karp, Richard M. (1972).
Reducibility among Combinatorial Problem, pages 85–103.
Springer, Boston, MA.

[Prestwich, 2003] Prestwich, Steven (2003).
Sat problems with chains of dependent variables.
Discrete Applied Mathematics, 130(2):329–350.
The Renesse Issue on Satisfiability.

[Sinz, 2005] Sinz, Carsten (2005).
Towards an optimal CNF encoding of Boolean cardinality constraints.
In van Beek, Peter, editor, *Proc. of 10th International Conference on Principles and Practice of Constraint Programming (CP 2005)*, pages 827–831, Berlin, Heidelberg. Springer.

[Soh+, 2014] Soh, Takehide, Le Berre, Daniel, Roussel, Stéphanie, Banbara, Mutsunori , and Tamura, Naoyuki (2014).
Incremental sat-based method with native boolean cardinality handling for the hamiltonian cycle problem.
In Fermé, Eduardo and Leite, João, editors, *Logics in Artificial Intelligence*, pages 684–693, Cham. Springer International Publishing.

Reference III

[Velev+, 2009] Velev, Miroslav N. and Gao, Ping (2009).
Efficient sat techniques for relative encoding of permutations with constraints.
In Nicholson, Ann and Li, Xiaodong, editors, *AI 2009: Advances in Artificial Intelligence*, pages 517–527, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Zhou, 2020] Zhou, Neng-Fa (2020).
In pursuit of an efficient SAT encoding for the hamiltonian cycle problem.
In Simonis, Helmut, editor, *Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings*, volume 12333 of *Lecture Notes in Computer Science*, pages 585–602. Springer.

Appendix

Additional Experimental Results with Cardinality-CDCL

$\vert V\vert$	#inst.	nohint	+2loop	+asym	+merge	-merge	-asym	-2loop	all	VBS
~ 1000	171	171	171	171	171	171	171	171	171	171
~ 2000	165	165	165	165	165	165	165	165	165	165
~ 3000	177	175	175	175	176	175	175	176	175	176
~ 4000	185	166	169	166	176	169	180	176	181	181
~ 5000	128	86	99	86	104	99	118	104	118	119
~ 6000	80	36	57	36	49	59	77	49	77	77
~ 7000	55	25	36	25	29	36	46	29	46	46
~ 8000	28	11	14	11	14	14	16	14	16	16
~ 9000	10	6	6	6	5	6	6	5	6	6
~ 10000	2	2	2	2	2	2	2	2	2	2
Total	1001	843	894	843	891	896	956	891	957	959

Encoding of Degree Constraints

To encode an exact-one constraint, both at-most-one and at-least-one constraints are added.

Encoding of the at-most-one constraint $x + y + z \leq 1$

$$\neg x \vee \neg y$$
$$\neg y \vee \neg z$$
$$\neg z \vee \neg x$$

Encoding of the at-least-one constraint $x + y + z \geq 1$

$$x \vee y \vee z$$

In practice, the Sinz encoding is used for at-most-one constraints.

Application to TSP

- Although direct application is difficult, for example, since many instances have degree 3, one possible approach is to extract the three highest-weight edges for each vertex to construct a degree-3 subgraph, solve HCP on it, and then optimize the subgraph.

Hardness of the Hamiltonian Cycle Problem

- Instances with very few Hamiltonian cycles relative to graph size are known to be difficult.
- Problems with repetitive structures or many low-degree vertices (degree 2 or less) are also difficult.

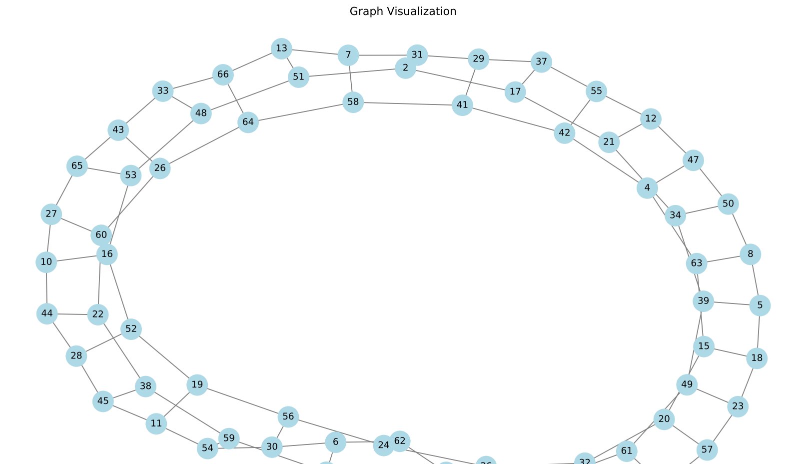
Features of Unsatisfied Instances in This Study

- Graphs where most vertices have degree between 3 and 6.
- Instances derived from change ringing.

About the Benchmark Instances

Instances in the FHCP Challenge were designed to be unsolvable by at least two out of three TSP-based solvers.

Smallest Instance



What is Change Ringing?

Overview

- A traditional technique of ringing multiple bells in specific sequences.
- Ringers change the order of bell ringing based on mathematical permutations.

Application to HCP

Each bell sequence is treated as a vertex in a graph. An edge is drawn between two sequences if one can be transformed into the other by a single operation. The problem can then be formulated as an HCP.

Each vertex represents a specific bell order, and each edge represents a reachable sequence by one operation (e.g., adjacent swap).

A Hamiltonian cycle on this graph corresponds to an exhaustive performance pattern covering all possible permutations in change ringing.

Proposed CEGAR: Hint Constraints and Additional Processing

Constraint to Forbid 2-Vertex Loops

For every arc $(i, j) \in A$:

$$\bigwedge_{(i,j)\in A} x_{ij} \rightarrow \neg x_{ji} \tag{1}$$

Constraint to Eliminate Symmetry

Let i be the vertex with the lowest degree. For all arc pairs $(i, j), (k, i) \in A$ with $j > k$:

$$\neg s_{i,j} \vee \neg s_{k,i} \tag{2}$$

Experimental Results: Evaluation of Hint Constraints and Processing (Sinz Encoding)

頂点数	問題数	nohint	+2loop	+asym	+merge	-merge	-asym	-2loop	all	VBS
~ 1000	171	171	171	171	170	171	171	171	171	171
~ 2000	165	165	165	162	165	165	165	163	164	165
~ 3000	177	173	177	172	175	174	175	173	175	177
~ 4000	185	152	166	151	158	166	166	159	167	169
~ 5000	128	86	101	90	96	107	118	98	118	119
~ 6000	80	35	52	37	47	55	76	51	75	77
~ 7000	55	20	35	23	27	37	44	29	44	47
~ 8000	28	9	11	9	13	13	15	12	15	16
~ 9000	10	4	4	5	5	5	5	5	5	5
~ 10000	2	2	2	2	2	2	2	2	2	2
合計	1001	817	884	822	858	895	937	863	936	948

eeded.

Experimental Results: Evaluation of Hint Constraints and Processing (Cardinality-CDCL)

$\lvert V \rvert$	#inst.	nohint	+2loop	+asym	+merge	-merge	-asym	-2loop	all	VBS
~ 1000	171	171	171	171	171	171	171	171	171	171
~ 2000	165	165	165	165	165	165	165	165	165	165
~ 3000	177	175	175	175	176	175	175	176	175	176
~ 4000	185	166	169	166	176	169	180	176	181	181
~ 5000	128	86	99	86	104	99	118	104	118	119
~ 6000	80	36	57	36	49	59	77	49	77	77
~ 7000	55	25	36	25	29	36	46	29	46	46
~ 8000	28	11	14	11	14	14	16	14	16	16
~ 9000	10	6	6	6	5	6	6	5	6	6
~ 10000	2	2	2	2	2	2	2	2	2	2
合計	1001	843	894	843	891	896	956	891	957	959

Experimental Environment and Benchmarks

- We implemented both the existing and proposed SAT-based CEGAR solvers in Rust. Other solvers were evaluated using publicly available source code.
- Hint Constraints and Processing in the Proposed SAT-based CEGAR:
  - 2loop: Prohibits 2-vertex loops.
  - asym: Symmetry elimination.
  - merge: Cycle merging strategy.
- Benchmark:
  - HCP instances (1,001 total) from the Flinders Hamiltonian Cycle Project (FHCP) [Haythorpe, 2019]
  - All instances are SAT (i.e., Hamiltonian cycles exist).
- CPU: 2.5GHz
- Memory: 64GB
- SAT Solver: CaDiCaL version 1.9.4
- Time Limit: 30 minutes per instance

Constraints of the Hamiltonian Cycle Problem (Directed)

V is a set of n vertices, A is the set of directed edges. $G = (V, A)$ is a directed graph.

$x_{ij} = 1 \Leftrightarrow$ the directed edge $(i, j) \in A$ is included in the solution.

Degree Constraints

$$\sum_{(i,j)\in A} x_{ij} = 1 \qquad \text{for each } i = 1, \ldots, n \quad \text{(out-degree)} \tag{3}$$

$$\sum_{(i,j)\in A} x_{ij} = 1 \qquad \text{for each } j = 1, \ldots, n \quad \text{(in-degree)} \tag{4}$$

Connectivity Constraint

$$\sum_{i,j\in S} x_{ij} \leq |S| - 1 \qquad S \subset V,\ 2 \leq |S| \leq n - 2 \tag{5}$$

- Naively encoding the connectivity constraint into SAT results in a very large number of clauses — proportional to n^3 in the number of vertices n.

Symmetry Elimination

- Let u be the vertex with the lowest (or highest) degree, and let $v = 1..n\ (u \neq v)$ be the vertices adjacent to u.
- Ensure that the vertex entering u has a smaller index than the one leaving u.

Example Constraint

$$\neg x_{un} \wedge \bigwedge_{i,j \in \{1...(n-1)\}, i<j} (\neg x_{iu} \vee \neg x_{uj}) \tag{6}$$

Alternative Formulation

$$\neg x_{un} \wedge \bigvee_{i \in \{2...n\}} x_{iu} \wedge \bigwedge_{j \in \{2...(n-1)\}} (\neg x_{uj} \vee \bigvee_{i>j} x_{iu}) \tag{7}$$

FHCP Challenge Results

The winning solver in the FHCP Challenge solved 985 out of 1,001 instances. The following 16 instances remained unsolved:
788, 868, 937, 951, 954, 960, 965, 966, 974, 976, 981, 983, 987, 993, 994, 998