# A SAT-based Method for Counting All Singleton Attractors in Boolean Networks

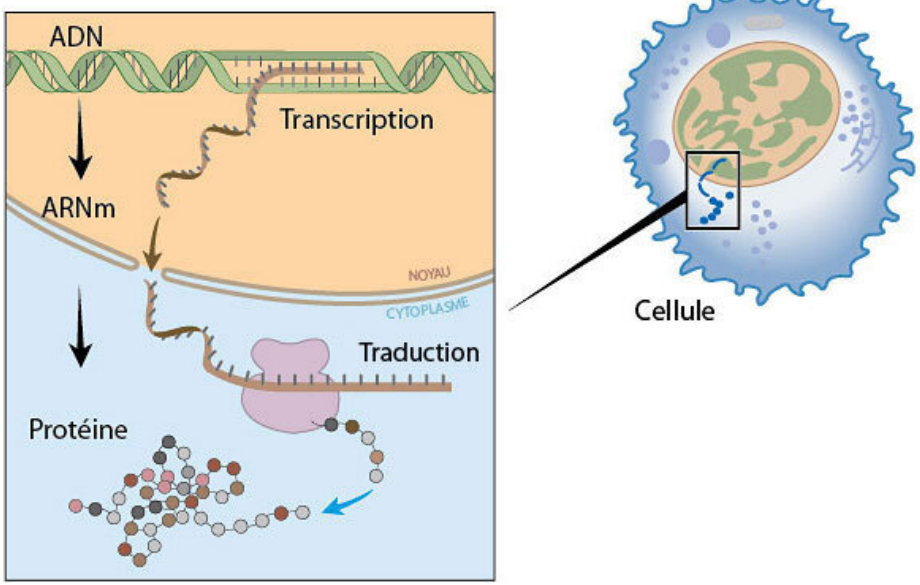
Rei Higuchi[1], Takehide Soh[1], Daniel Le Berre[2], Morgan Magnin[3], Mutsunori Banbara[4], Naoyuki Tamura[1]

[1]Kobe University,
[2]CRIL-CNRS UMR 8188, Université d'Artois,
[3]LS2N, UMR 6004,
[4]Nagoya University

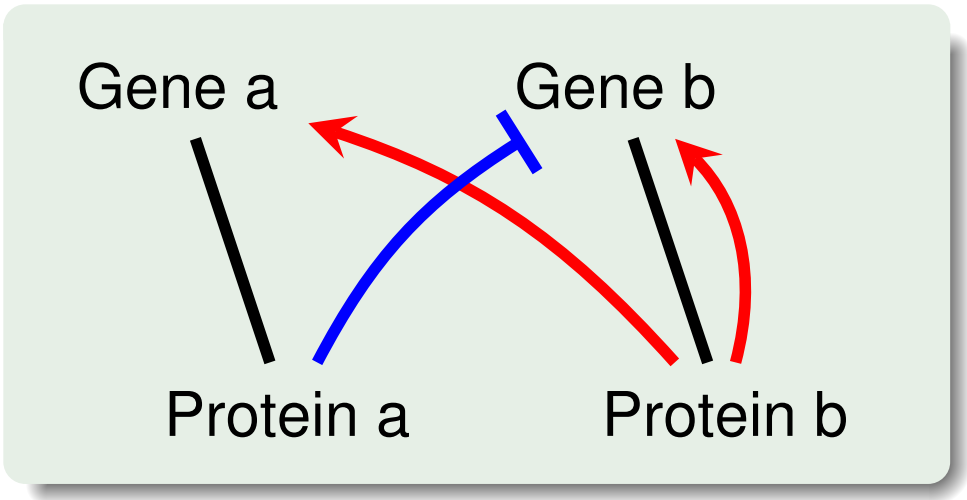
IJCAI 2025 Montreal
Aug. 21 2025

# Gene Regulatory Networks

- Living organisms use genetic information to produce proteins, which shape their structure and functions.
- Some proteins regulate the expression of their own genes or those of others.




The Nobel Committee for Physiology or Medicine / Illustration: Annika Röhl




- A **gene regulatory network** shows how genes and proteins interact.

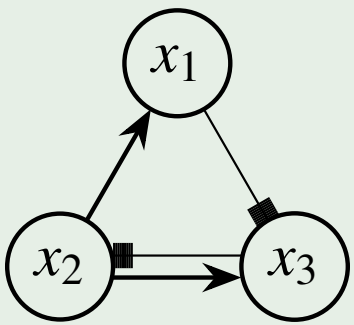
# Boolean Network [Kauffman, 1969]

- A **Boolean Network** (BN) is a representative mathematical model for describing gene regulatory networks.
- Each gene is represented by a propositional variable $x_i$, and its expression state is expressed as a binary value.
- The regulatory function associated with each gene is represented by a propositional formula $\psi_i$.

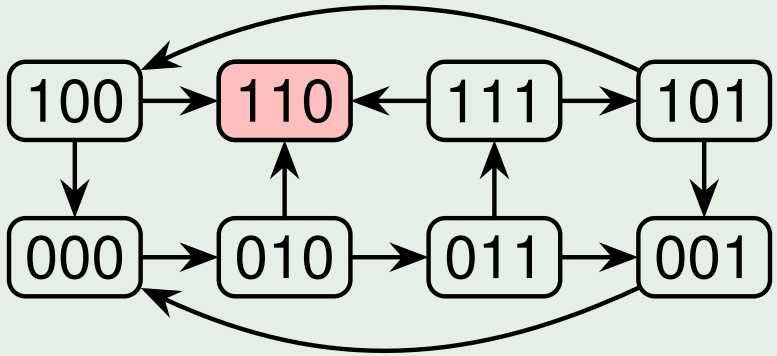
# Boolean Network [Kauffman, 1969]

- A **Boolean Network** (BN) is a representative mathematical model for describing gene regulatory networks.
- Each gene is represented by a propositional variable $x_i$, and its expression state is expressed as a binary value.
- The regulatory function associated with each gene is represented by a propositional formula $\psi_i$.

## Example of a BN (given as a set of equations $x_i^{t+1} = \psi_i^t$)




$$x_1^{t+1} = x_2^t$$
$$x_2^{t+1} = \neg x_3^t$$
$$x_3^{t+1} = \neg x_1^t \wedge x_2^t$$




- This BN has $2^3$ possible states.
- **An attractor** is a stable state of a gene regulatory network, in the sense that once the system enters it, it cannot escape.

# Challenges in Attractor Analysis

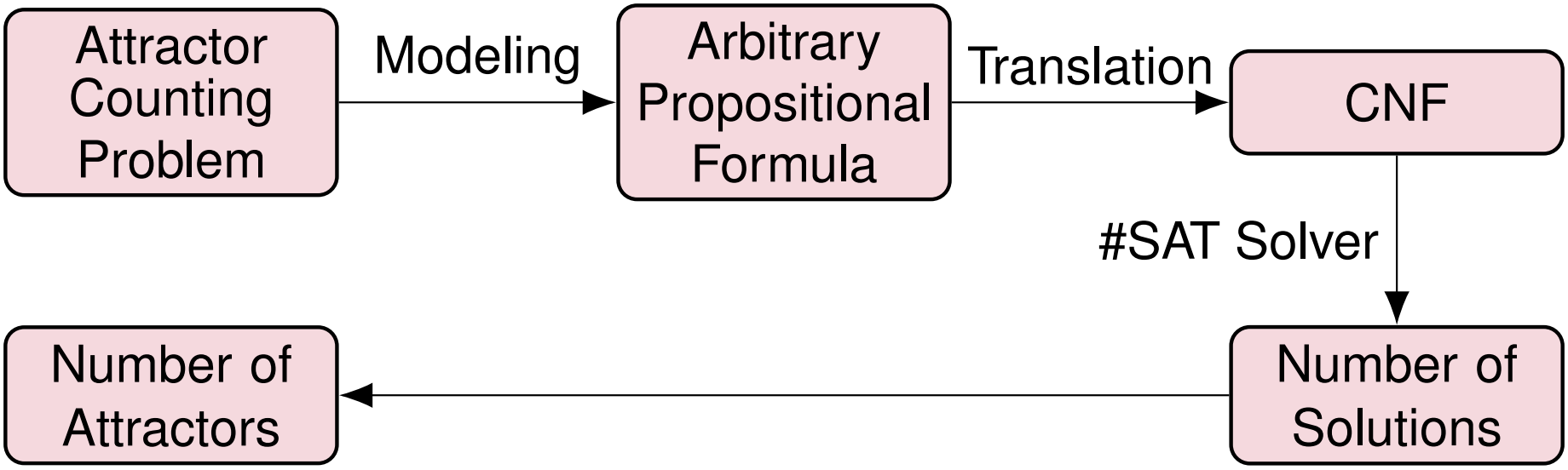
- BNs are collected and published in repositories aimed at the formal verification of biological knowledge:
  - **GINsim** [Naldi+, 2018]
  - **CellCollective** [Helikar+, 2012]
  - **Biomodels** [Malik-Sheriff+, 2020]
- Many BN analysis tools have also been proposed:
  - **Boolsim** [Garg+, 2008], **Pyboolnet** [Klarner+, 2017], **CABEAN** [Su+, 2021], **pystablemotifs** [Rozum+, 2022], **AEON** [Benes+, 2022], **SAF** [Soh+, 2023], **fASP** [Trinh+, 2023]

However, in recent years, some BNs published in these repositories have contained more than $300$ genes and over $10^{20}$ attractors.
For such large-scale networks, existing analysis tools have struggled not only to enumerate attractors, but even to count them.

- Counting is important for evaluating the roles of genes in a BN.

# Research Objective and Content

Develop a **SAT-based BN Attractor Counting Solver** for fast attractor counting.




## Research Content

- **Constraint Modeling**: An arbitrary propositional formula whose solutions correspond to attractors
- **CNF translation**: A method to convert propositional formulas into CNF with as few variables and clauses as possible

# Arbitrary Propositional Formula Whose Solutions Correspond to Attractors

## Input of the Attractor Counting Problem ($n$ = number of genes)

$$x_i^{t+1} = \psi_i^t \qquad (i = 1, \ldots, n)$$

- Condition for $x_i$ **to change from** $0$ **to** $1$ between $t$ and $t+1$

$$\neg x_i^t \wedge \psi_i^t$$

- Condition for $x_i$ **to change from** $1$ **to** $0$ between $t$ and $t+1$

$$x_i^t \wedge \neg \psi_i^t$$

For state $t = 0$ to be a attractor, no transitions other than self-loops may occur. This leads to the following condition:

## Propositional Formula Whose Models Correspond to Attractors

$$\bigwedge_{1 \leq i \leq n} \neg(\neg x_i^0 \wedge \psi_i^0) \wedge \neg(x_i^0 \wedge \neg \psi_i^0)$$

# Translation to CNF

$$\Psi = \bigwedge_{1 \leq i \leq n} \neg(\neg x_i \wedge \psi_i) \wedge \neg(x_i \wedge \neg \psi_i)$$

- Although the proposed constraint model is a propositional formula, in a BN each $\psi_i$ is given in an arbitrary form, so it is not in CNF.
- To use existing counting solvers efficiently, it is necessary to convert the formula into CNF with as few variables and clauses as possible.

## Proposed CNF translation

We propose the following three types of translation:

1. Direct translation (using Tseitin transformation)
2. Indirect translation (using model enumeration)
3. Hybrid translation

# Direct translation

## Before translation

$$\Psi = \bigwedge_{1 \leq i \leq n} \neg(\neg x_i \wedge \psi_i) \wedge \neg(x_i \wedge \neg \psi_i)$$

- The **Tseitin transformation** [Tseitin, 1968] is a method for converting an arbitrary propositional formula into a satisfiability-equivalent CNF by recursively introducing new variables that are satisfiability-equivalent to subformulas.
- Instead of applying it directly to $\Psi$, we apply it to each $\psi_i$ to obtain a satisfiability-equivalent literal $l_i$.

## CNF After translation

$$\Psi_D = \bigwedge_{1 \leq i \leq n} \neg(\neg x_i \wedge l_i) \wedge \neg(x_i \wedge \neg l_i) \wedge \mathit{TseitinTrans}(l_i \leftrightarrow \psi_i)$$

# Example of Direct translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation

$$\Psi_D = \neg(\neg x_1 \wedge l_1) \wedge \neg(x_1 \wedge \neg l_1) \wedge TseitinTrans(l_1 \leftrightarrow \psi_1)$$

# Example of Direct translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation

$$\begin{aligned}\Psi_D &= \neg(\neg x_1 \wedge l_1) \wedge \neg(x_1 \wedge \neg l_1) \wedge TseitinTrans(l_1 \leftrightarrow \psi_1) \\ &= \neg(\neg x_1 \wedge l_1) \wedge \neg(x_1 \wedge \neg l_1) \wedge\end{aligned}$$

# Example of Direct translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation

$$\begin{aligned}\Psi_D &= \neg(\neg x_1 \wedge l_1) \wedge \neg(x_1 \wedge \neg l_1) \wedge TseitinTrans(l_1 \leftrightarrow \psi_1) \\ &= \neg(\neg x_1 \wedge l_1) \wedge \neg(x_1 \wedge \neg l_1) \wedge \\ &\quad (p_1 \leftrightarrow x_2 \vee x_3 \vee x_4) \wedge\end{aligned}$$

# Example of Direct translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation

$$\begin{aligned}\Psi_D &= \neg(\neg x_1 \wedge l_1) \wedge \neg(x_1 \wedge \neg l_1) \wedge TseitinTrans(l_1 \leftrightarrow \psi_1) \\ &= \neg(\neg x_1 \wedge l_1) \wedge \neg(x_1 \wedge \neg l_1) \wedge \\ &\quad (p_1 \leftrightarrow x_2 \vee x_3 \vee x_4) \wedge \\ &\quad (p_2 \leftrightarrow x_2 \vee \neg x_5) \wedge\end{aligned}$$

# Example of Direct translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation

$$\begin{aligned}
\Psi_D &= \neg(\neg x_1 \wedge l_1) \wedge \neg(x_1 \wedge \neg l_1) \wedge TseitinTrans(l_1 \leftrightarrow \psi_1) \\
&= \neg(\neg x_1 \wedge l_1) \wedge \neg(x_1 \wedge \neg l_1) \wedge \\
&\quad (p_1 \leftrightarrow x_2 \vee x_3 \vee x_4) \wedge \\
&\quad (p_2 \leftrightarrow x_2 \vee \neg x_5) \wedge \\
&\quad (l_1 \leftrightarrow p_1 \wedge p_2)
\end{aligned}$$

# Example of Direct translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation

$$\Psi_D = \neg(\neg x_1 \wedge l_1) \wedge \neg(x_1 \wedge \neg l_1) \wedge TseitinTrans(l_1 \leftrightarrow \psi_1)$$

$$= \neg(\neg x_1 \wedge l_1) \wedge \neg(x_1 \wedge \neg l_1) \wedge$$

$$(p_1 \leftrightarrow x_2 \vee x_3 \vee x_4) \wedge$$
$$(p_2 \leftrightarrow x_2 \vee \neg x_5) \wedge$$
$$(l_1 \leftrightarrow p_1 \wedge p_2)$$

Tseitin Transformation

# Indirect translation

## Before translation

$$\Psi = \bigwedge_{1 \leq i \leq n} \neg(\neg x_i \wedge \psi_i) \wedge \neg(x_i \wedge \neg\psi_i)$$

- Convert $\neg x_i \wedge \psi_i$ and $x_i \wedge \neg\psi_i$ into DNF form.
- Since the negation of a DNF is a CNF, the CNF can be obtained as follows.

## CNF After translation

$$\Psi_I = \bigwedge_{1 \leq i \leq n} \neg DNF(\neg x_i \wedge \psi_i) \wedge \neg DNF(x_i \wedge \neg\psi_i)$$

- Let $DNF(\phi) = \bigvee_{m \in \mathcal{M}(\phi)} \bigwedge_{l \in m} l$, where $\mathcal{M}$ is the set of all complete solutions.
- Since the clauses in $DNF(\phi)$ are long and can be extremely numerous, we aim to compute a logically equivalent but more compact $DNF^P(\phi)$.

# Computation Method for $DNF^P(\phi)$

**Algorithm 1** Enumeration of All Partial Models $DNF^P(\phi)$

**Input**: Propositional formula $\phi$, upper bound on #models $co$
**Output**:Full enumeration possible; if so, partial models $\mathcal{M}^P$

1: $\Omega :=$ Apply Tseitin transformation (one-way) to $\phi$
2: $\Omega_{PI} :=$ Transform $\Omega$ according to [Jabbour+, 2014]
3: $(\text{isOK}, M)$ := **ModelEnumeration**$(\Omega_{PI}, co)$ ▷ Enumerates PIs of $\Omega$
4: **Return** $(\text{isOK}, \{m \in M \mid m \cap Var(\phi)\})$

- The prime implicants of $\Omega$ do not coincide with those of $\phi$, but the $DNF^P(\phi)$ obtained by this algorithm is logically equivalent to $DNF(\phi)$ and yields a set of partial solutions that is significantly smaller than the set of complete solutions.

**Note**

$DNF^P(\phi)$ generates far fewer and shorter models than $DNF(\phi)$.

# Example of Indirect translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation: Case of $DNF(\phi)$

$$\Psi_I = \neg DNF(\neg x_1 \wedge \psi_1) \wedge \neg DNF(x_1 \wedge \neg \psi_1)$$

# Example of Indirect translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation: Case of $DNF(\phi)$

$$\begin{aligned}
\Psi_I &= \neg DNF(\neg x_1 \wedge \psi_1) \wedge \neg DNF(x_1 \wedge \neg \psi_1) \\
&= \neg(\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5) \ \wedge \ \neg(\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge \neg x_5) \wedge \\
&\quad \neg(\neg x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4 \wedge x_5) \ \wedge \ \neg(\neg x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4 \wedge \neg x_5) \wedge \\
&\quad \neg(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4 \wedge x_5) \ \wedge \ \neg(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4 \wedge \neg x_5) \wedge \\
&\quad \neg(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4 \wedge x_5) \ \wedge \ \neg(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4 \wedge \neg x_5) \wedge \\
&\quad \neg(\neg x_1 \wedge \neg x_2 \wedge x_3 \wedge x_4 \wedge \neg x_5) \ \wedge \ \neg(\neg x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4 \wedge \neg x_5) \wedge \\
&\quad \neg(\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4 \wedge \neg x_5) \ \wedge
\end{aligned}$$

The number of complete solutions of $DNF(\neg x_1 \wedge \psi_1)$ is 11.

# Example of Indirect translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation: Case of $DNF(\phi)$

$$\begin{aligned}
\Psi_I &= \neg DNF(\neg x_1 \wedge \psi_1) \wedge \neg DNF(x_1 \wedge \neg \psi_1) \\
&= \neg(\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5) \ \wedge \ \neg(\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge \neg x_5) \wedge \\
&\quad \neg(\neg x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4 \wedge x_5) \ \wedge \ \neg(\neg x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4 \wedge \neg x_5) \wedge \\
&\quad \neg(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4 \wedge x_5) \ \wedge \ \neg(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4 \wedge \neg x_5) \wedge \\
&\quad \neg(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4 \wedge x_5) \ \wedge \ \neg(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4 \wedge \neg x_5) \wedge \\
&\quad \neg(\neg x_1 \wedge \neg x_2 \wedge x_3 \wedge x_4 \wedge \neg x_5) \ \wedge \ \neg(\neg x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4 \wedge \neg x_5) \wedge \\
&\quad \neg(\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4 \wedge \neg x_5) \ \wedge \ \neg DNF(x_1 \wedge \neg \psi_1)
\end{aligned}$$

The number of complete solutions of $DNF(\neg x_1 \wedge \psi_1)$ is 11.

# Example of Indirect translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation: Case of $DNF^P(\phi)$

$$\Psi_I = \neg DNF^P(\neg x_1 \wedge \psi_1) \wedge \neg DNF^P(x_1 \wedge \neg \psi_1)$$

# Example of Indirect translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation: Case of $DNF^P(\phi)$

$$\begin{aligned}\Psi_I &= \neg DNF^P(\neg x_1 \wedge \psi_1) \wedge \neg DNF^P(x_1 \wedge \neg\psi_1) \\ &= \neg(\neg x_1 \wedge x_2) \ \wedge \ \neg(\neg x_1 \wedge x_3 \wedge \neg x_5) \ \wedge \ \neg(\neg x_1 \wedge x_4 \wedge \neg x_5) \wedge \\ &\quad \neg DNF^P(x_1 \wedge \neg\psi_1)\end{aligned}$$

The number of solutions of $DNF^P(\neg x_1 \wedge \psi_1)$ is 3.

# Example of Indirect translation

## Before translation

$$x_1^{t+1} = (x_2^t \vee x_3^t \vee x_4^t) \wedge (x_2^t \vee \neg x_5^t)$$

## CNF After translation: Case of $DNF^P(\phi)$

$$\begin{aligned}\Psi_I &= \neg DNF^P(\neg x_1 \wedge \psi_1) \wedge \neg DNF^P(x_1 \wedge \neg \psi_1) \\ &= \neg(\neg x_1 \wedge x_2) \ \wedge \ \neg(\neg x_1 \wedge x_3 \wedge \neg x_5) \ \wedge \ \neg(\neg x_1 \wedge x_4 \wedge \neg x_5) \wedge \\ &\quad \neg(x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4) \ \wedge \ \neg(x_1 \wedge \neg x_2 \wedge x_5)\end{aligned}$$

In this example, the total number of solutions is reduced from $16$ to $5$.

# Hybrid translation

- The Direct translation has the advantage of not requiring model enumeration, while the Indirect translation has the advantage of not requiring new variables. We propose a translation method that combines the strengths of both.

---

**Algorithm 2** Hybrid translation

---

**Input**: Propositional variable $x_i$, update function $\psi_i$, threshold $cutoff$
**Output**: CNF

1: $(\mathrm{isOK}^+, M^+)$ := **Enumeration of small partial solutions**$(\neg x_i \wedge \psi_i, cutoff)$
2: **if** $\neg\mathrm{isOK}^+$ **then return** $(x_i \vee \neg l_i) \wedge (\neg x_i \vee l_i) \wedge TT(l_i \leftrightarrow \psi_i)$ ▷ Direct
3: $(\mathrm{isOK}^-, M^-)$ := **Enumeration of small partial solutions**$(x_i \wedge \neg\psi_i, cutoff)$
4: **if** $\mathrm{isOK}^-$ **then return** $\neg(M^+ \cup M^-)$ ▷ Indirect
5: **return** $(x_i \vee \neg l_i) \wedge (\neg x_i \vee l_i) \wedge TT(l_i \leftrightarrow \psi_i)$ ▷ Direct

---

- In this study, we set $cutoff = 1000$ based on preliminary exp.

# Experiments

## Benchmark Problems

Experiments were conducted using all 643 BNs from [Trinh+, 2023].

- Real-world: BBM (230), Selected (13)
- Synthetic: P-Random (400)

## Comparison Methods

- Proposed methods:
  - Direct, Indirect, Hybrid + SharpSAT-TD [Korhonen+, 2023]
  - Hybrid + BDD_MINISAT_ALL [Toda+, 2016]
- Existing methods:
  - PyBoolNet [Klarner+, 2017]
  - SAF [Soh+, 2023]
  - fASP-c, fASP-s [Trinh+, 2023]
  - AEON [Benes+, 2022]

# Number of Problems Solved by Each Method

## A) Number of problems solved in each benchmark set

| | # | Existing Tools | | | | Proposed Methods | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | SAF | fASP-c | fASP-s | AEON | H.E. | D.C. | I.C. | H.C. |
| BBM | 230 | 220 | 195 | 213 | 224 | 218 | **230** | 229 | **230** |
| P-Random | 400 | 0 | 15 | 15 | 12 | 15 | 71 | **88** | **88** |
| Selected | 13 | 6 | 6 | 7 | 8 | 7 | **12** | 11 | **12** |
| Total | 643 | 226 | 216 | 235 | 244 | 240 | 313 | 328 | **330** |

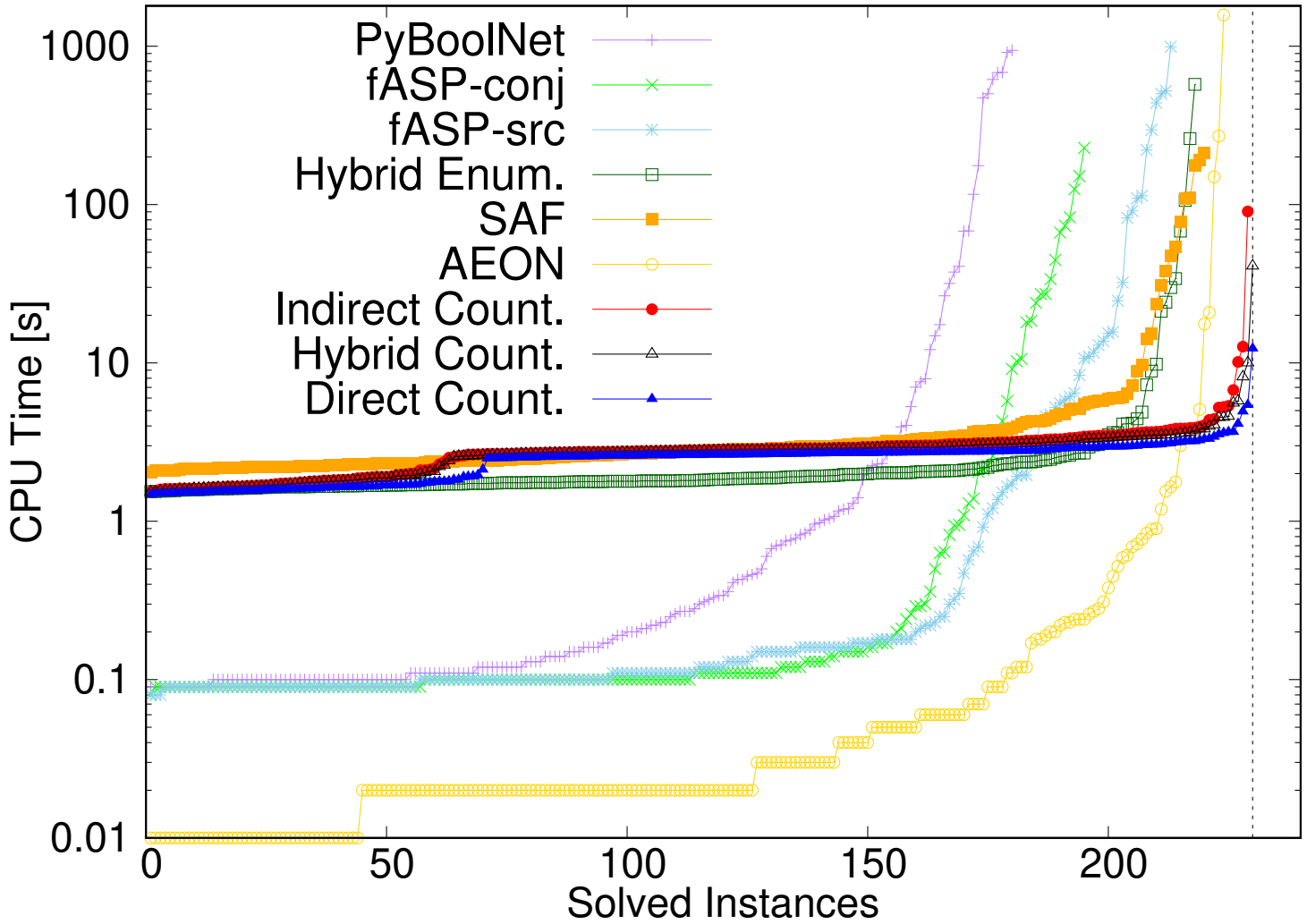
## B) Number of problems solved by range of attractor counts $|A|$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $0 \leq \|A\| < 1000$ | 157 | 142 | **157** | **157** | 154 | **157** | **157** | **157** | **157** |
| $1000 \leq \|A\| < 10^{10}$ | 72 | 70 | 59 | 70 | **72** | 71 | **72** | **72** | **72** |
| $10^{10} \leq \|A\| < 10^{30}$ | 27 | 14 | 0 | 8 | 18 | 12 | **27** | 25 | **27** |
| $10^{30} \leq \|A\|$ | 387 | 0 | 0 | 0 | 0 | 0 | 57 | **74** | **74** |

## C) Number of problems solved by range of gene count $n$

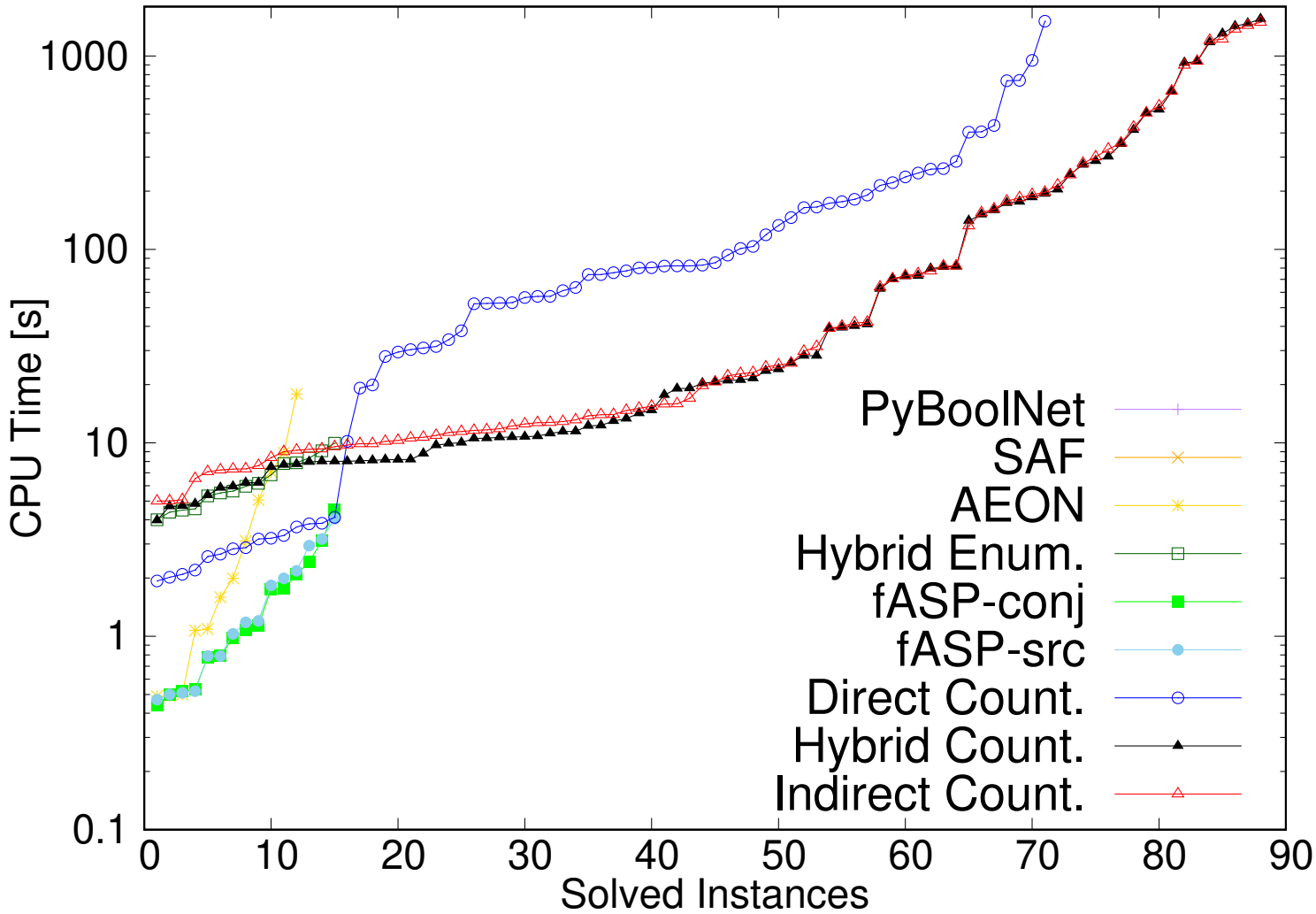
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $0 \leq n < 100$ | 198 | 196 | 182 | 194 | **198** | 197 | **198** | **198** | **198** |
| $100 \leq n < 1000$ | 44 | 30 | 19 | 26 | 34 | 28 | **44** | 42 | **44** |
| $1000 \leq n < 2000$ | 111 | 0 | 6 | 6 | 6 | 6 | 41 | **49** | **49** |
| $2000 \leq n$ | 290 | 0 | 9 | 9 | 6 | 9 | 30 | **39** | **39** |

# Cactus Plot: BBM (230)



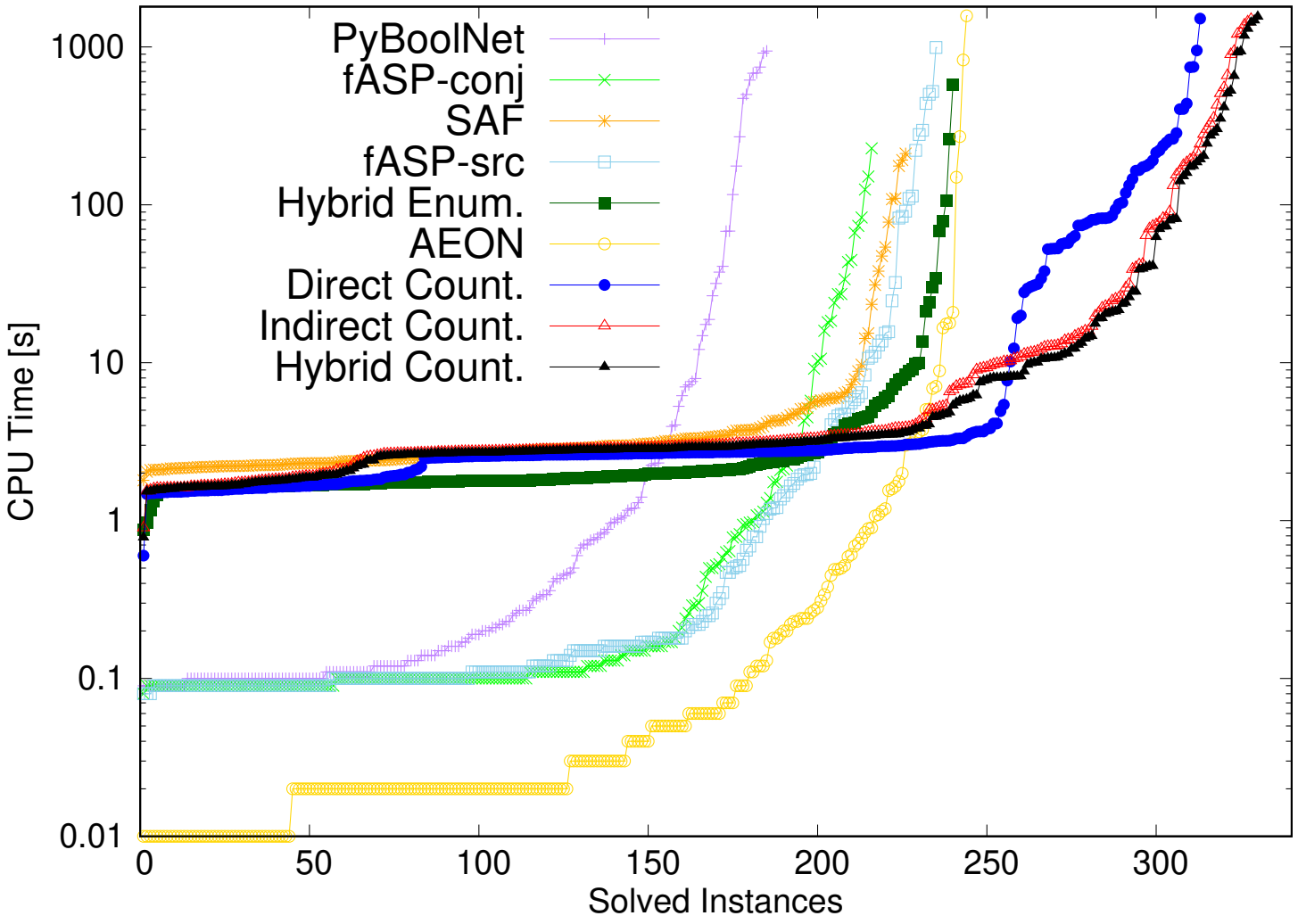

Only the proposed method was able to determine the number of attractors for all BNs.

# Cactus Plot: P-Random (400)




Only the proposed method solved more than 15 instances.

# Cactus Plot (CPU Time)




Only the proposed method solved more than 250 instances.

# Comparison of CNFs in the Proposed Method

| Benchmark Set | Timeout | | | #Variables | | #Clauses | | #Literals | |
|---|---|---|---|---|---|---|---|---|---|
| | D | I | H | I | H | I | H | I | H |
| BBM | 0 | 1 | 0 | 0.47 | 0.48 | 0.49 | 0.49 | 0.58 | 0.58 |
| P-Random | 0 | 0 | 0 | 0.22 | 0.22 | 0.52 | 0.52 | 1.52 | 1.52 |
| Selected | 0 | 2 | 0 | 0.36 | 0.36 | 0.68 | 0.61 | 1.90 | 1.34 |

# Problems with Previously Unknown Numbers of Attractors

- Across 400 synthetic BNs, counted fixed-points for 88; 75 previously-unknown.
- Across 243 real-world BNs, counted fixed-points for 242; 9 previously-unknown as shown below.

| Instance Name | #Genes | #Attractors |
|---|---|---|
| #113 ER-STRESS | 182 | 1168455003694263561093120 |
| #122 NSP14 | 168 | 33278627362665583108034953216 |
| #124 NSP9-PROTEIN | 252 | 136112946768375385385349842972707284582 40 |
| #144 SNF1-AMPK-PATHWAY | 202 | 10096027719780900754667077632 |
| #220 H.-RESPONSE-IN-L. | 342 | 2656331146614175432704000 |
| #221 MYCOBACTERIAL-L. | 317 | 2473901162496 |
| **Cell-Cycle-Control** | **3158** | **—** |
| Alzheimer | 762 | 135531809447440039244514002058631920 9- -710396035433027073714342803602931712 0 |
| Cholocystokinin | 383 | 47935169240579835005239296 |
| *Leishmania (same as #220 above)* | *342* | *2656331146614175432704000* |
| Yeast-Pheromone | 246 | 5711631030629640192 |

# Summary

- We proposed a method for counting attractors in BNs, a mathematical model of gene regulatory networks.
- **Contributions of this study**
  - Proposed a constraint model and CNF translation for the attractor counting problem.
  - Successfully counted attractors that existing tools could not handle (9 real-world, 75 synthetic).

## Future Directions

- Verify the biological significance of attractor analysis results obtained from counting.
- Comparative evaluation of CNF translation methods for propositional formulas.
- Apply SAT-based methods to other BN problems such as cyclic attractors, bifurcation, and basins of attraction.

# Supplemental Slides

# Significance of Counting Attractors

## Significance in Gene Regulatory Networks

- The effect of operations such as single knockout (SKO) or single overexpression on the network can be analyzed by counting attractors (see next slide).
- Serves as one indicator when modeling gene regulatory networks.

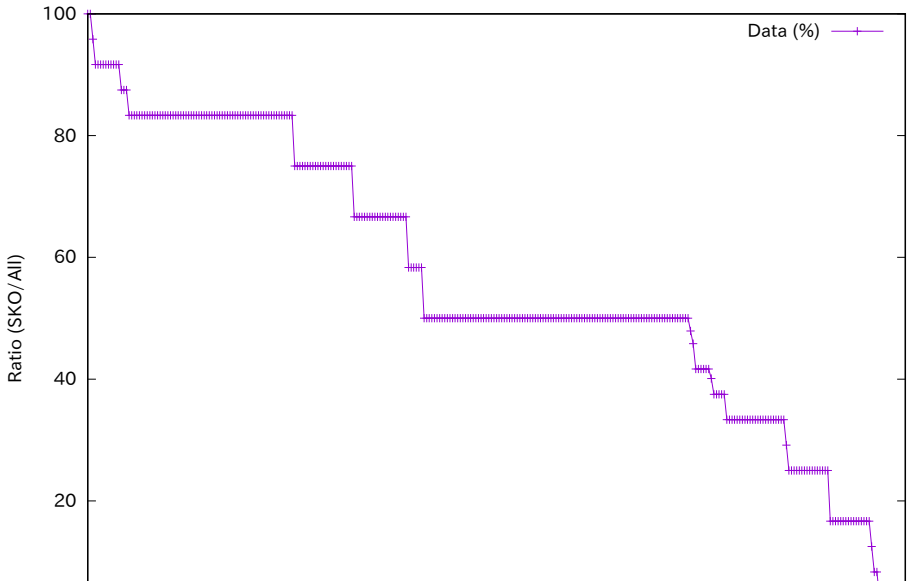
## Significance in Computing Attractors

- Serves as a reference for determining whether attractor enumeration is feasible.
- When enumerating a very large number of attractors in parallel, counting can be used as an indicator for load balancing.

# #221 MYCOBACTERIAL-L. [Ganguli+ 2012]

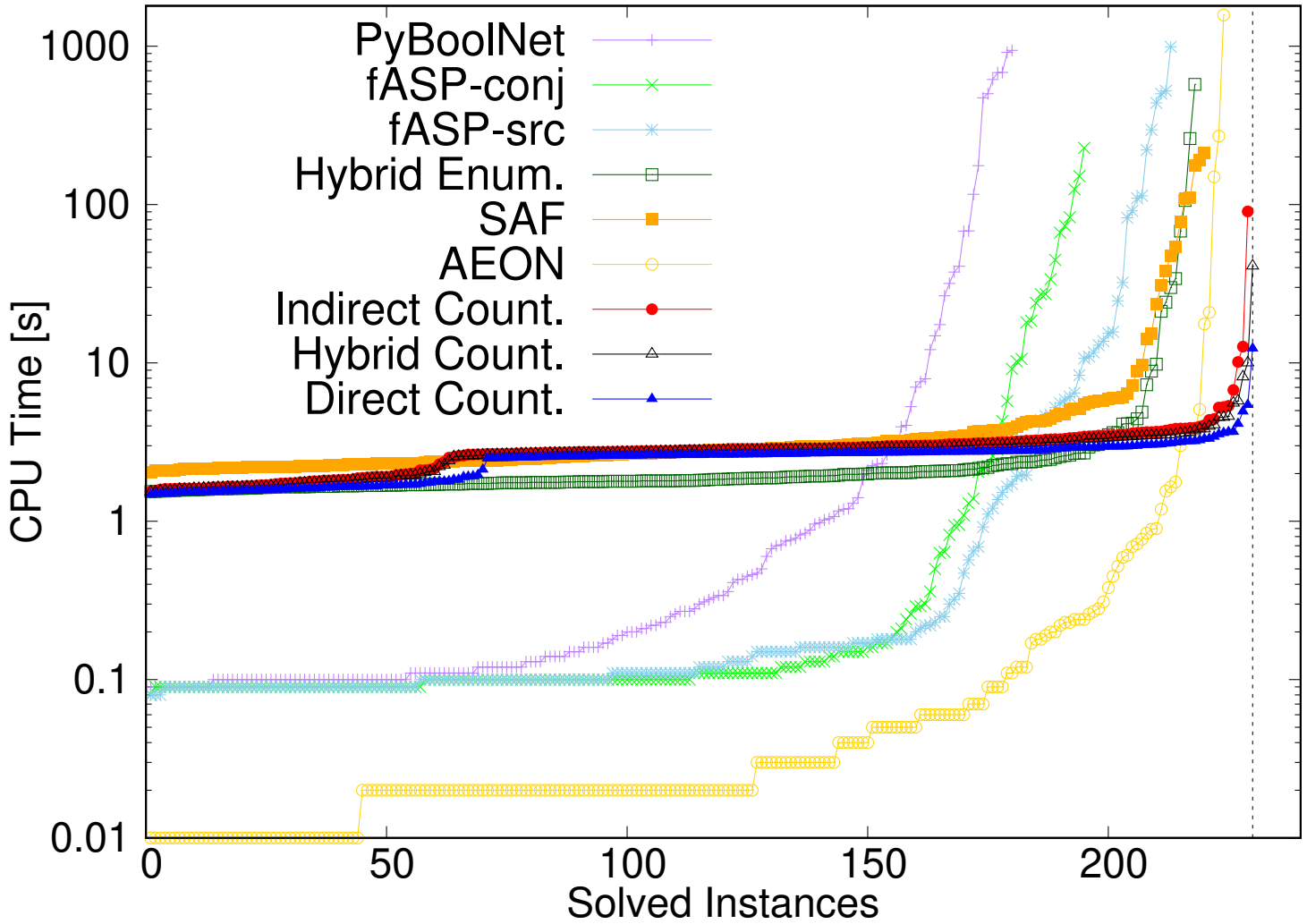
This BN centers on a group of genes that change specifically during the dormancy phase of *Mycobacterium tuberculosis*, incorporating the transcription factors regulating them and their functional interactions. Stable states are interpreted as dormancy states.

**Single knockout (SKO)** refers to the genetic operation of inactivating or deleting a specific single gene in an organism's genome. This technique is used to study how a specific gene affects development, metabolism, and disease.



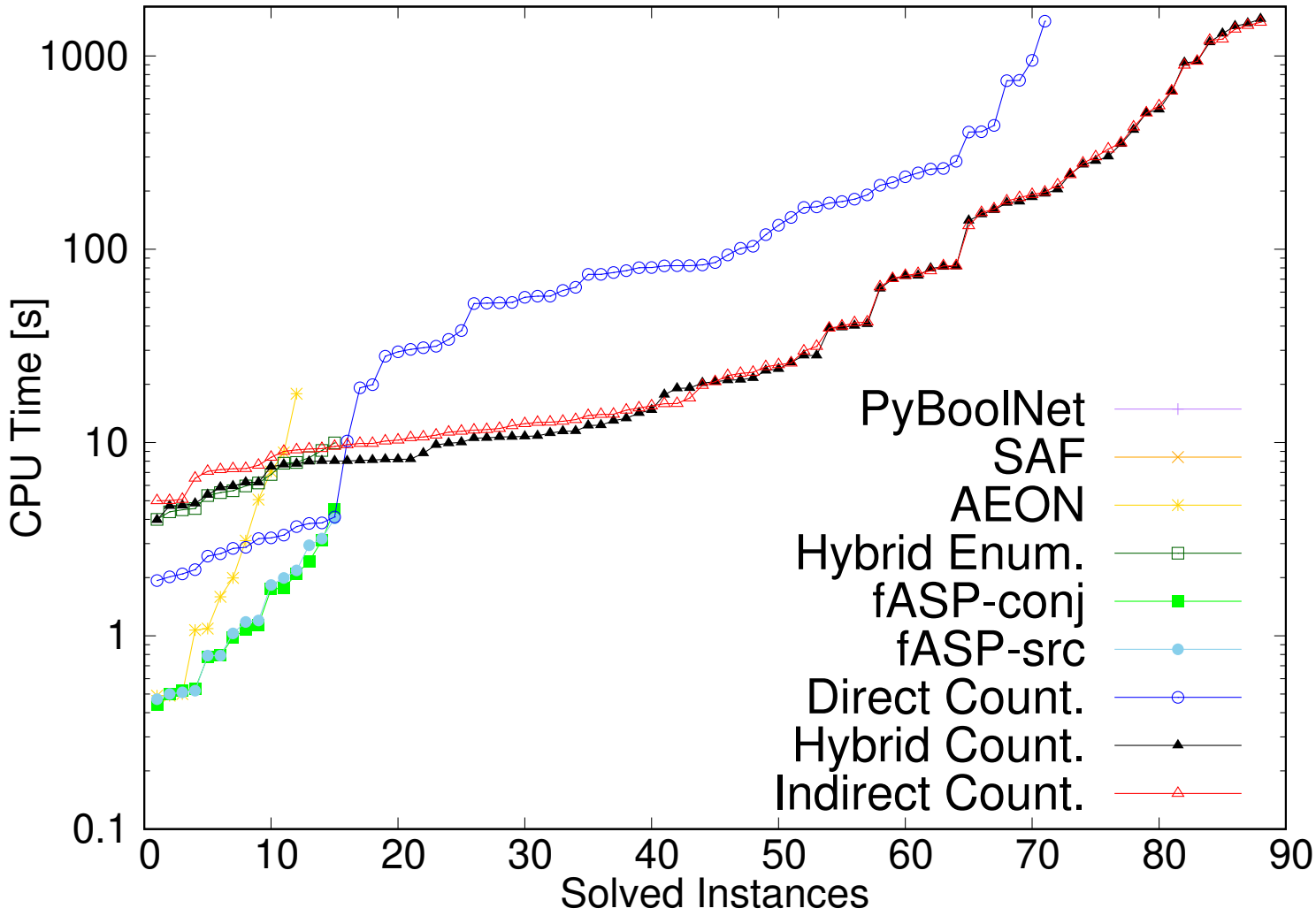

The impact of SKO on stable states can be analyzed via counting. In the figure, the SKO of the rightmost gene reduces the number of attractors to 0 (no stable states), rendering the network unstable. One such gene is Rv3132c (DosS), which agrees with the literature

# Cactus Plot: BBM (230)



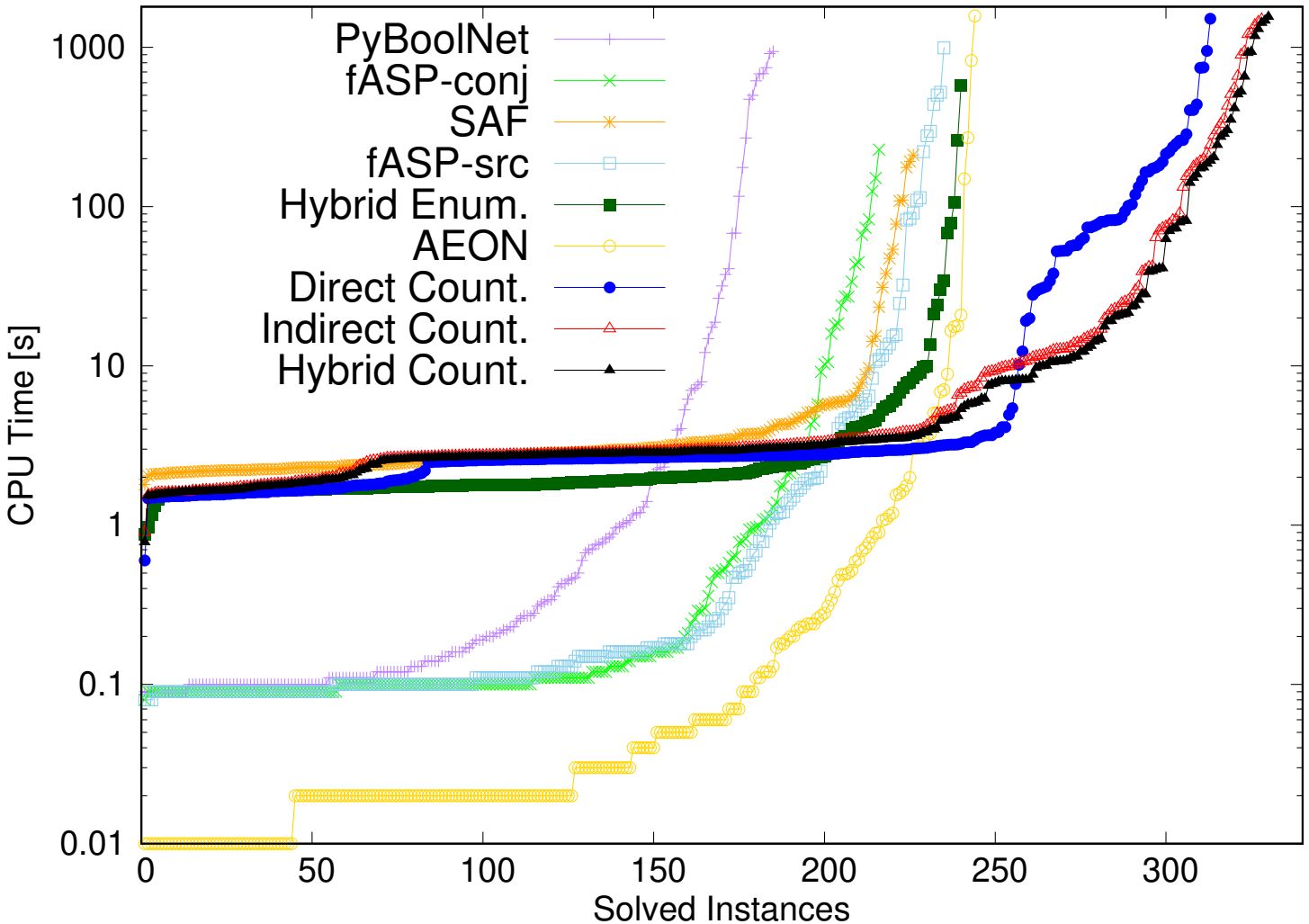

Only the proposed method was able to determine the number of attractors for all BNs.

# Cactus Plot: P-Random (400)




Only the proposed method solved more than 15 instances.

# Cactus Plot (CPU Time)

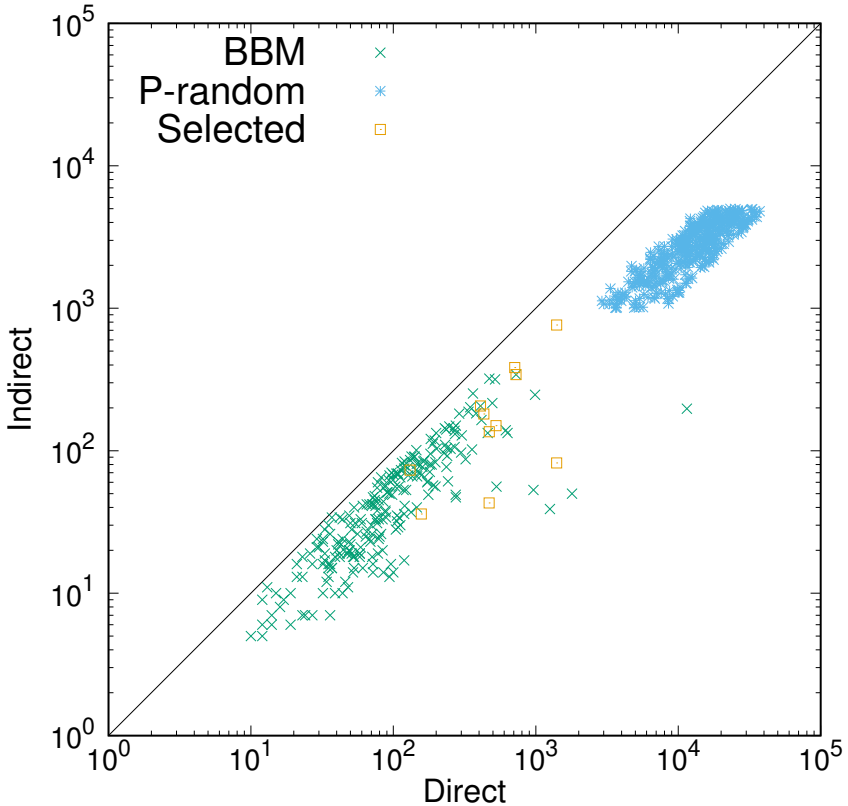


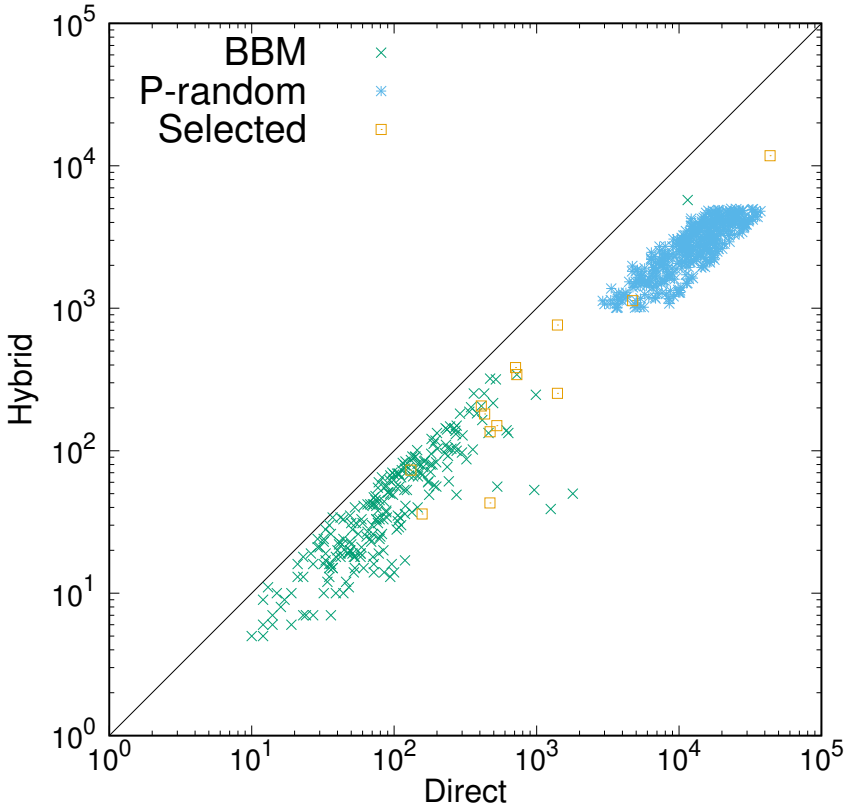

Only the proposed method solved more than 250 instances.

# Comparison of CNFs in the Proposed Method

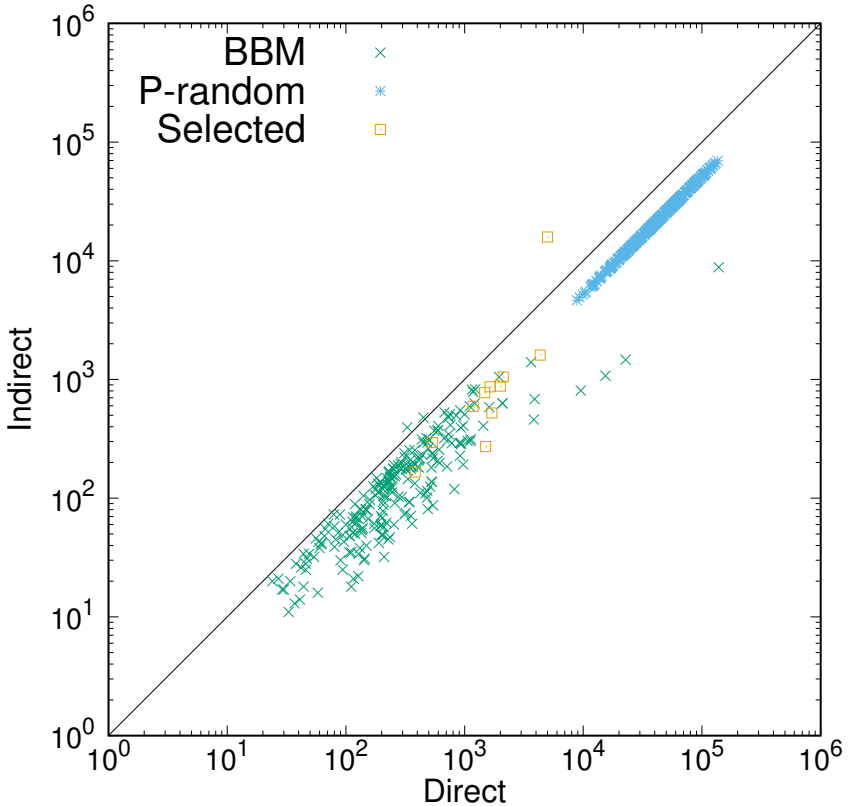
| Benchmark Set | Timeout | | | #Variables | | #Clauses | | #Literals | |
|---|---|---|---|---|---|---|---|---|---|
| | D | I | H | I | H | I | H | I | H |
| BBM | 0 | 1 | 0 | 0.47 | 0.48 | 0.49 | 0.49 | 0.58 | 0.58 |
| P-Random | 0 | 0 | 0 | 0.22 | 0.22 | 0.52 | 0.52 | 1.52 | 1.52 |
| Selected | 0 | 2 | 0 | 0.36 | 0.36 | 0.68 | 0.61 | 1.90 | 1.34 |

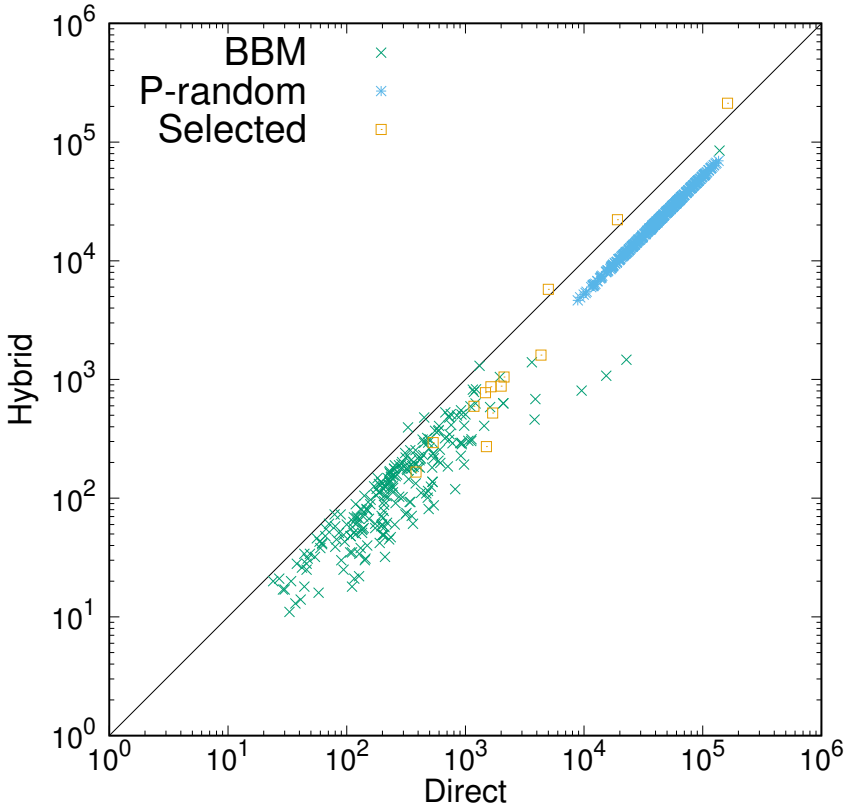
# Scatter Plot (Number of Variables)



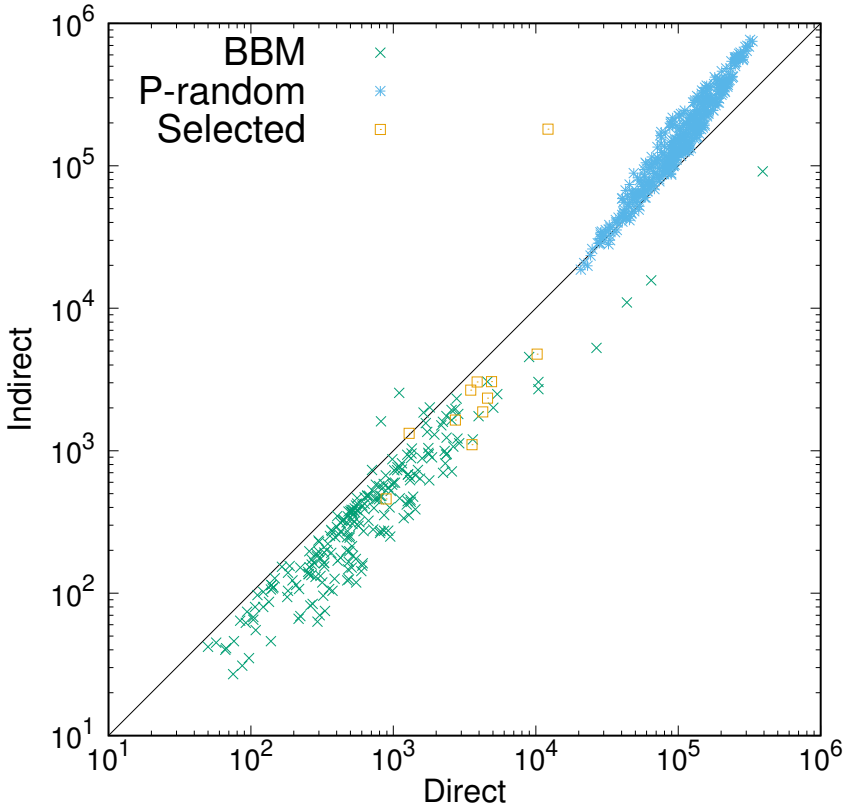



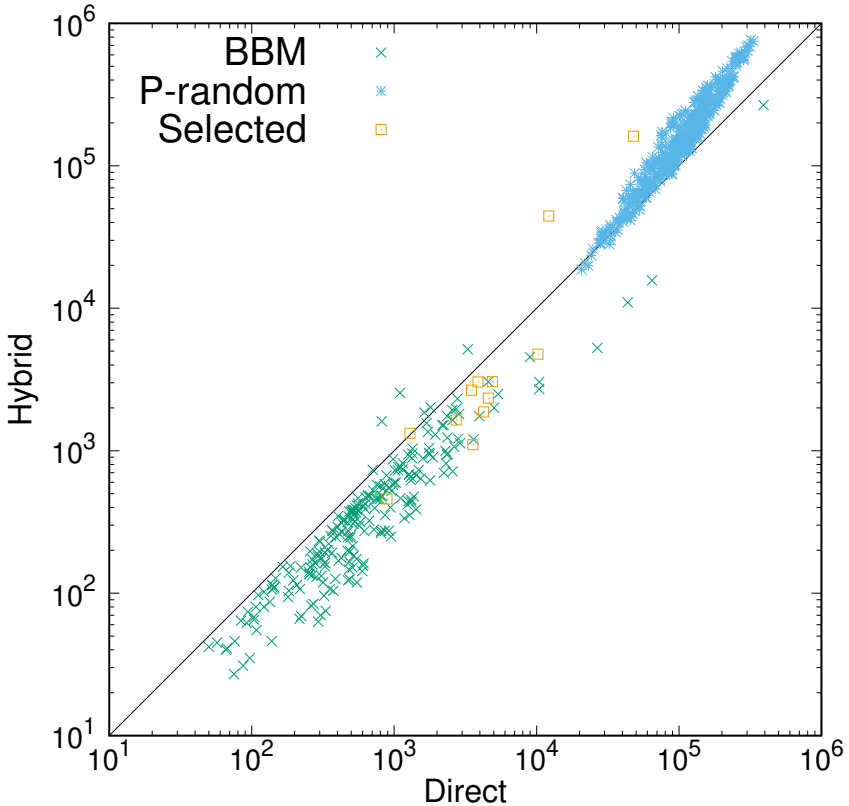
# Scatter Plot (Number of Clauses)



# Scatter Plot (Number of Literals)



# Limitations of Existing Methods

## Existing Methods

Various studies have been conducted to find attractors:

- **Boolsim** [Garg+, 2008]
- **Pyboolnet** [Klarner+, 2017]
- **CABEAN** [Su+, 2021]
- **pystablemotifs** [Rozum+, 2022]
- **AEON** [Benes+, 2022]
- **SAF** [Soh+, 2023]
- **fASP** [Trinh+, 2023]

However, there exist BNs for which these existing methods cannot find attractors.

# Limitations of Enumeration-based Existing Methods

- All existing methods for finding attractors use enumeration.
- Recently published BNs may contain extremely large numbers of attractors ($\geq 10^{20}$), making enumeration infeasible for these methods.

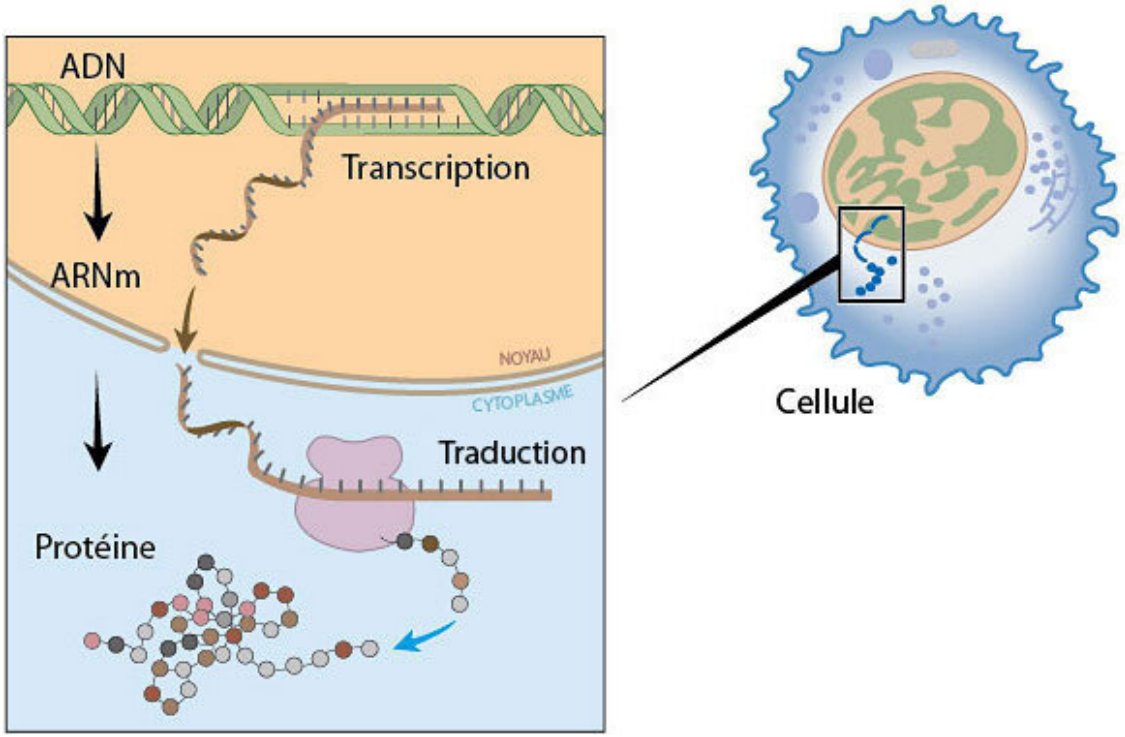
Knowing the number of attractors is useful for evaluating the importance of specific genes in a BN. Therefore, counting instead of enumeration is a promising approach.

## Can Existing Tools Be Used as Counting Solvers?

- Some recent enumeration methods provide an option to return only the number of attractors to reduce I/O costs.
- However, many problems could not be solved even when outputs were suppressed.

# Genome, Genes, and Proteins

- A **genome** is the genetic information of an organism, consisting of a sequence of bases A/T/G/C.
- A **gene** is a subsequence of the genome.
- All living organisms produce **proteins** based on the information in their genes, forming their structure and functions to sustain life.




The Nobel Committee for Physiology or Medicine / Illustration: Annika Röhl

- Proteins can regulate the expression of their own genes and those

# Number of Instances Solved by Each Method

A) Number of instances solved in each benchmark set

| | | Existing Methods | | | | | Proposed Methods | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | #Ins. | PyB. | SAF | fASP-c | fASP-s | AEON | H.E. | D.C. | I.C. | H.C. |
| BBM | 230 | 180 | 220 | 195 | 213 | 224 | 218 | **230** | 229 | **230** |
| P-Random | 400 | 0 | 0 | 15 | 15 | 12 | 15 | 71 | **88** | **88** |
| Selected | 13 | 5 | 6 | 6 | 7 | 8 | 7 | **12** | 11 | **12** |
| Total | 643 | 185 | 226 | 216 | 235 | 244 | 240 | 313 | 328 | **330** |

B) Number of instances solved by range of attractor counts $|A|$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $0 \leq \|A\| < 1000$ | 157 | 138 | 142 | **157** | **157** | 154 | **157** | **157** | **157** | **157** |
| $1000 \leq \|A\| < 10^{10}$ | 72 | 47 | 70 | 59 | 70 | **72** | 71 | **72** | **72** | **72** |
| $10^{10} \leq \|A\| < 10^{30}$ | 27 | 0 | 14 | 0 | 8 | 18 | 12 | **27** | 25 | **27** |
| $10^{30} \leq \|A\|$ | 387 | 0 | 0 | 0 | 0 | 0 | 0 | 57 | **74** | **74** |

C) Number of instances solved by range of variable count $n$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $0 \leq n < 100$ | 198 | 170 | 196 | 182 | 194 | **198** | 197 | **198** | **198** | **198** |
| $100 \leq n < 1000$ | 44 | 15 | 30 | 19 | 26 | 34 | 28 | **44** | 42 | **44** |
| $1000 \leq n < 2000$ | 111 | 0 | 0 | 6 | 6 | 6 | 6 | 41 | **49** | **49** |
| $2000 \leq n$ | 290 | 0 | 0 | 9 | 9 | 6 | 9 | 30 | **39** | **39** |

# Reference

# Reference I


Benes, Nikola, Brim, Lubos, Huvar, Ondrej, Pastva, Samuel, Safránek, David , and Smijáková, Eva (2022).
Aeon.py: Python library for attractor analysis in asynchronous boolean networks.
Bioinform., 38(21):4978–4980.

Garg, Abhishek, Di Cara, Alessandro, Xenarios, Ioannis, Mendoza, Luis , and De Micheli, Giovanni (2008).
Synchronous versus asynchronous modeling of gene regulatory networks.
Bioinformatics, 24(17):1917–1925.

# Reference II


Jabbour, Saïd, Marques-Silva, João, Sais, Lakhdar , and Salhi, Yakoub (2014).
Enumerating prime implicants of propositional formulae in conjunctive normal form.
In Fermé, Eduardo and Leite, João, editors, Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings, volume 8761 of Lecture Notes in Computer Science, pages 152–165. Springer.

Kauffman, Stuart A. (1969).
Metabolic stability and epigenesis in randomly constructed genetic nets.
Journal of Theoretical Biology, 22(3):437–467.

# Reference III



Klarner, Hannes, Streck, Adam , and Siebert, Heike (2017).
Pyboolnet: a python package for the generation, analysis and visualization of boolean networks.
Bioinform., 33(5):770–772.


Korhonen, Tuukka and Järvisalo, Matti (2023).
Sharpsat-td in model counting competitions 2021-2023.


Rozum, Jordan C., Deritei, Dávid, Park, Kyu Hyong, Zañudo, Jorge Gómez Tejeda , and Albert, Réka (2022).
pystablemotifs: Python library for attractor identification and control in boolean networks.
Bioinform., 38(5):1465–1466.

# Reference IV


Soh, Takehide, Magnin, Morgan, Berre, Daniel Le, Banbara, Mutsunori , and Tamura, Naoyuki (2023).
Sat-based method for finding attractors in asynchronous multi-valued networks.
In Ali, Hesham, Deng, Ning, Fred, Ana L. N. , and Gamboa, Hugo, editors, Proceedings of the 16th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2023, Volume 3: BIOINFORMATICS, Lisbon, Portugal, February 16-18, 2023, pages 163–174. SCITEPRESS.

Su, Cui and Pang, Jun (2021).
CABEAN: a software for the control of asynchronous boolean networks.
Bioinform., 37(6):879–881.

# Reference V


Toda, Takahisa and Soh, Takehide (2016).
Implementing efficient all solutions SAT solvers.
ACM J. Exp. Algorithmics, 21(1):1.12:1–1.12:44.

Trinh, Van-Giang, Benhamou, Belaid , and Soliman, Sylvain (2023).
Efficient enumeration of fixed points in complex boolean networks using answer set programming.
In Yap, Roland H. C., editor, 29th International Conference on Principles and Practice of Constraint Programming, CP 2023, August 27-31, 2023, Toronto, Canada, volume 280 of LIPIcs, pages 35:1–35:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.