Scalable Counting of Minimal Trap Spaces and Fixed Points in Boolean Networks

Mohimenul Kabir, Van-Giang Trinh, Samuel Pastva, Kuldeep S. Meel

Boolean Network (BN)

A Boolean Network $\mathcal{N} = (V, F)$, where

- $V = \{v_1, \ldots, v_n\}$ is a set of nodes
- $F = \{f_1, \ldots, f_n\}$ is a set of associated Boolean functions

Boolean Network (BN)

A Boolean Network $\mathcal{N} = (V, F)$, where

- $V = \{v_1, \dots, v_n\}$ is a set of nodes
- $F = \{f_1, \dots, f_n\}$ is a set of associated Boolean functions

Update Scheme: At time t, node $v_i \in V$ can update its state by $s_{t+1}(v_i) = f_i(s_t)$.

- Asynchronous: exactly one node updated at each time step *non-deterministically*.
- Synchronous: all nodes updated at each time step.

Boolean Network (BN)

A Boolean Network $\mathcal{N} = (V, F)$, where

- $V = \{v_1, \ldots, v_n\}$ is a set of nodes
- $F = \{f_1, \ldots, f_n\}$ is a set of associated Boolean functions

Update Scheme: At time t, node $v_i \in V$ can update its state by $s_{t+1}(v_i) = f_i(s_t)$.

- Asynchronous: exactly one node updated at each time step *non-deterministically*.
- Synchronous: all nodes updated at each time step.

$$V = \{a, b\}$$

$$\begin{cases} f_a = (a \wedge \neg b) \\ f_b = a \end{cases}$$

Boolean network $\mathcal{N}$

$$\begin{tikzcd} 00 \arrow[loop left] & 01 \arrow[l] \\ 10 \arrow[r] & 11 \arrow[u] \end{tikzcd}$$

STG($\mathcal{N}$)

Trap spaces in BN

A *sub-space* is a map $m : V \mapsto \{0, 1, \star\}$ representing a state hypercube.

- $0\star \sim \{00, 01\}$
- $11 \sim \{11\}$
- $\star\star \sim \{00, 01, 10, 11\}$

where, $\star$ denotes the variable is *free*; otherwise the variable is *fixed*.

Trap spaces in BN

A *sub-space* is a map $m : V \mapsto \{0, 1, \star\}$ representing a state hypercube.

- $0\star \sim \{00, 01\}$
- $11 \sim \{11\}$
- $\star\star \sim \{00, 01, 10, 11\}$

where, $\star$ denotes the variable is *free*; otherwise the variable is *fixed*.

- Trap Space: A sub-space that is a set of states from which the system cannot escape once entered.
- Minimal Trap Space (MTS): A trap space that has no other smaller trap spaces.
- Fixed Point (FIX): A special case of minimal trap space where no variable is free.

Notably, trap spaces are independent of the employed update scheme [KBS2015].

Trap spaces in BN

A *sub-space* is a map $m : V \mapsto \{0, 1, \star\}$ representing a state hypercube.

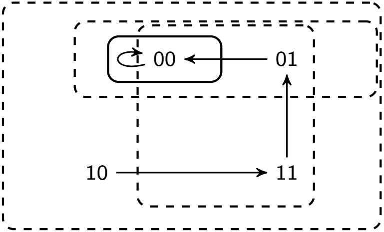
- $0\star \sim \{00, 01\}$
- $11 \sim \{11\}$
- $\star\star \sim \{00, 01, 10, 11\}$

where, $\star$ denotes the variable is *free*; otherwise the variable is *fixed*.

- Trap Space: A sub-space that is a set of states from which the system cannot escape once entered.
- Minimal Trap Space (MTS): A trap space that has no other smaller trap spaces.
- Fixed Point (FIX): A special case of minimal trap space where no variable is free.

$$\begin{cases} f_a = (a \wedge \neg b) \\ f_b = a \end{cases}$$

Boolean network $\mathcal{N}$



STG($\mathcal{N}$)

Some Definitions

Definition (Phenotype)

A *trait* is a statement of form: $(v \longleftrightarrow e)$, where $v \in \mathrm{Var}(\mathcal{N})$ and $e \in \{0, 1, \star\}$.
A *phenotype* β is the conjunction of a set of traits, $\beta \equiv \bigwedge_i (v_i \longleftrightarrow e_i)$.

Some Definitions

Definition (Phenotype)

A *trait* is a statement of form: $(v \longleftrightarrow e)$, where $v \in \mathrm{Var}(\mathcal{N})$ and $e \in \{0, 1, \star\}$.
A *phenotype* β is the conjunction of a set of traits, $\beta \equiv \bigwedge_i (v_i \longleftrightarrow e_i)$.

Definition (Perturbation)

A *perturbation* σ over a set of *perturbable* variables $\mathcal{X} \subseteq \mathrm{Var}(\mathcal{N})$ is a mapping as $\mathcal{X} \mapsto \{0, 1, \star\}$. For each variable $v \in \mathrm{Var}(\mathcal{N})$,

$$\sigma(v) = \begin{cases} f_v = 0 & \text{knockout} \\ f_v = 1 & \text{over-expression} \\ f_v = \star & f_v \text{ is unchanged} \end{cases}$$

Some Definitions

Definition (Phenotype)

A *trait* is a statement of form: $(v \longleftrightarrow e)$, where $v \in \mathrm{Var}(\mathcal{N})$ and $e \in \{0, 1, \star\}$.
A *phenotype* β is the conjunction of a set of traits, $\beta \equiv \bigwedge_i (v_i \longleftrightarrow e_i)$.

Definition (Perturbation)

A *perturbation* σ over a set of *perturbable* variables $\mathcal{X} \subseteq \mathrm{Var}(\mathcal{N})$ is a mapping as $\mathcal{X} \mapsto \{0, 1, \star\}$. For each variable $v \in \mathrm{Var}(\mathcal{N})$,

$$\sigma(v) = \begin{cases} f_v = 0 & \text{knockout} \\ f_v = 1 & \text{over-expression} \\ f_v = \star & f_v \text{ is unchanged} \end{cases}$$

Definition (Perturbed BN)

Given a perturbation σ, the *perturbed Boolean Network* $\mathcal{N}^\sigma = (V^\sigma, F^\sigma)$ where $V^\sigma = V$ and for each variable $v \in \mathcal{N}$,

$$f_v^\sigma = \begin{cases} \sigma(v) & \text{if } v \in \mathcal{X} \text{ and } \sigma(v) \neq \star \\ f_v & \text{otherwise} \end{cases}$$

Answer Set Programming (ASP)

- Roots in logic programming and non-monotonic reasoning
- A rule-based language for problem encoding

$$\underbrace{h_1 \vee \dots h_\ell}_{head} \leftarrow \underbrace{b_1, \dots, b_k, \sim b_{k+1}, \dots, \sim b_{k+m}.}_{body}$$

Answer Set Programming (ASP)

- Roots in logic programming and non-monotonic reasoning
- A rule-based language for problem encoding

$$\underbrace{h_1 \vee \ldots h_\ell}_{head} \leftarrow \underbrace{b_1, \ldots, b_k, \sim b_{k+1}, \ldots, \sim b_{k+m}.}_{body}$$

- An ASP program $P \equiv$ set of rules.
- Definitions:
  - Program P is called *disjunctive* if $\exists r \in P$ s.t. $|\text{Head}(r)| > 1$ [EG95]
  - Otherwise, program P is called *normal*
- The model of P is an *answer set* (denoted as AS(P)).

Answer Set Programming (ASP)

- Roots in logic programming and non-monotonic reasoning
- A rule-based language for problem encoding

$$\underbrace{h_1 \vee \ldots h_\ell}_{head} \leftarrow \underbrace{b_1, \ldots, b_k, \sim b_{k+1}, \ldots, \sim b_{k+m}.}_{body}$$

- An ASP program $P \equiv$ set of rules.
- Definitions:
  - Program P is called *disjunctive* if $\exists r \in P$ s.t. $|\mathrm{Head}(r)| > 1$ [EG95]
  - Otherwise, program P is called *normal*
- The model of P is an *answer set* (denoted as AS(P)).
- Answer set programming has close relationship with Systems Biology
- Existing and efficient trap spaces enumeration techniques rely on ASP and ASP solvers [KBS2015;PKC+2020;TBH+2023;TBS2023;TBP+2024].

New Challenges: Quantitative Reasoning on BN

⚠ Existing works focus on qualititive reasoning on BN (e.g., finding a trap space).

New Challenges: Quantitative Reasoning on BN

⚠ Existing works focus on qualititive reasoning on BN (e.g., finding a trap space).

💡 Interesting and biologically motivated research directions: Quantitative Reasoning

How to do Probabilistic Reasoning on BN?

New Challenges: Quantitative Reasoning on BN

⚠ Existing works focus on qualititive reasoning on BN (e.g., finding a trap space).

Interesting and biologically motivated research directions: Quantitative Reasoning

How to do Probabilistic Reasoning on BN?

How is the likelihood of a phenotype in a BN?

New Challenges: Quantitative Reasoning on BN

Existing works focus on qualititive reasoning on BN (e.g., finding a trap space).

Interesting and biologically motivated research directions: Quantitative Reasoning

How to do Probabilistic Reasoning on BN?

How is the likelihood of a phenotype in a BN?

How to quantify Robustness of a phenotype?

Our Contributions

We propose six meaningful counting problems on Boolean Networks.

	1st problems	2nd problems	3rd problems
Input	BN $\mathcal{N}$	BN $\mathcal{N}$, phenotype β	BN $\mathcal{N}$, phenotype β, perturbables $\mathcal{X}$

Our Contributions

We propose six meaningful counting problems on Boolean Networks.

	1st problems	2nd problems	3rd problems
Input	BN $\mathcal{N}$	BN $\mathcal{N}$, phenotype β	BN $\mathcal{N}$, phenotype β, perturbables $\mathcal{X}$
FIXs Formulation	C-FIX-1: #FIXs of $\mathcal{N}$	C-FIX-2: #FIXs of $\mathcal{N}$, satisfying β	C-FIX-3: #FIXs of $\mathcal{N}$, satisfying β, over perturbed BNs

Our Contributions

We propose six meaningful counting problems on Boolean Networks.

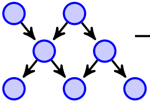
	1st problems	2nd problems	3rd problems
Input	BN $\mathcal{N}$	BN $\mathcal{N}$, phenotype β	BN $\mathcal{N}$, phenotype β, perturbables $\mathcal{X}$
FIXs Formulation	C-FIX-1: #FIXs of $\mathcal{N}$	C-FIX-2: #FIXs of $\mathcal{N}$, satisfying β	C-FIX-3: #FIXs of $\mathcal{N}$, satisfying β, over perturbed BNs
MTSs Formulation	C-MTS-1: #MTSs of $\mathcal{N}$	C-MTS-2: #MTSs of $\mathcal{N}$, satisfying β	C-MTS-3: #MTSs of $\mathcal{N}$, satisfying β, over perturbed BNs

Our Contributions

We propose six meaningful counting problems on Boolean Networks.

	1st problems	2nd problems	3rd problems
Input	BN $\mathcal{N}$	BN $\mathcal{N}$, phenotype β	BN $\mathcal{N}$, phenotype β, perturbables $\mathcal{X}$
FIXs Formulation	C-FIX-1: #FIXs of $\mathcal{N}$	C-FIX-2: #FIXs of $\mathcal{N}$, satisfying β	C-FIX-3: #FIXs of $\mathcal{N}$, satisfying β, over perturbed BNs
MTSs Formulation	C-MTS-1: #MTSs of $\mathcal{N}$	C-MTS-2: #MTSs of $\mathcal{N}$, satisfying β	C-MTS-3: #MTSs of $\mathcal{N}$, satisfying β, over perturbed BNs
Applications	probabilistic reasoning on BN	quantifying emergence of phenotype	phenotype robustness

Counting Methodologies: from high-level



Given $\mathcal{N}, \beta, \mathcal{X}$

$\longrightarrow$ $\mathcal{P} = \begin{array}{c} \star \leftarrow \star \\ \star \leftarrow \star \\ \star \leftarrow \star \end{array}$ $\longrightarrow$ $\#\mathsf{AS}(\mathcal{P})_{\downarrow\Delta}$

$\Delta = \{\ldots\}$

- The counting problems C-FIX-3 and C-MTS-3 reduce to *projected* answer set counting and the projection set Δ is derived from perturbable variables $\mathcal{X}$.
- For remaining counting problems, the projection set Δ is trivial.

Counting Formulation for C-FIX-3 and C-MTS-3

1. Capture all FIXs/MTSs

2. Satisfy Phenotype

3. Capture all FIXs/MTSs over perturbations

Capture all FIXs/MTSs (1/3)

- C-FIX-1: `fASP` [TBS2023] captures the fixed points
  the answer sets of the program one-to-one correspond with the FIXs of $\mathcal{N}$

Capture all FIXs/MTSs (1/3)

- C-FIX-1: `fASP` [TBS2023] captures the fixed points
  the answer sets of the program one-to-one correspond with the FIXs of $\mathcal{N}$
- C-MTS-1: `tsconj` [TBP+2024] captures the minimal trap spaces
  the answer sets of the program one-to-one correspond with the MTSs of $\mathcal{N}$

Capture all FIXs/MTSs (1/3)

- C-FIX-1: `fASP` [TBS2023] captures the fixed points
  the answer sets of the program one-to-one correspond with the FIXs of $\mathcal{N}$
- C-MTS-1: `tsconj` [TBP+2024] captures the minimal trap spaces
  the answer sets of the program one-to-one correspond with the MTSs of $\mathcal{N}$

ASP Encodings

$$\mathcal{P} = \begin{cases} \text{fASP}(\mathcal{N}) & \text{for all FIXs} \\ \text{tsconj}(\mathcal{N}) & \text{for all MTSs} \end{cases}$$

Capture all FIXs/MTSs (1/3)

- C-FIX-1: `fASP` [TBS2023] captures the fixed points
  the answer sets of the program one-to-one correspond with the FIXs of $\mathcal{N}$
- C-MTS-1: `tsconj` [TBP+2024] captures the minimal trap spaces
  the answer sets of the program one-to-one correspond with the MTSs of $\mathcal{N}$

ASP Encodings

$$\mathcal{P} = \begin{cases} \mathsf{fASP}(\mathcal{N}) & \text{for all FIXs} \\ \mathsf{tsconj}(\mathcal{N}) & \text{for all MTSs} \end{cases}$$

Some Notes on `tsconj` and `fASP`

- for each variable $v \in \mathsf{Var}(\mathcal{N})$, there are two atoms $\mathsf{p}(v)$ and $\mathsf{n}(v)$

Capture all FIXs/MTSs (1/3)

- C-FIX-1: `fASP` [TBS2023] captures the fixed points
  the answer sets of the program one-to-one correspond with the FIXs of $\mathcal{N}$
- C-MTS-1: `tsconj` [TBP$^+$2024] captures the minimal trap spaces
  the answer sets of the program one-to-one correspond with the MTSs of $\mathcal{N}$

ASP Encodings

$$\mathcal{P} = \begin{cases} \mathsf{fASP}(\mathcal{N}) & \text{for all FIXs} \\ \mathsf{tsconj}(\mathcal{N}) & \text{for all MTSs} \end{cases}$$

Some Notes on `tsconj` and `fASP`

- for each variable $v \in \mathsf{Var}(\mathcal{N})$, there are two atoms $\mathsf{p}(v)$ and $\mathsf{n}(v)$
- The relationship between answer set A of the program $\mathcal{P}$ and sub-space m of $\mathcal{N}$ is that for every variable $v \in \mathsf{Var}(\mathcal{N})$:
  - $m(v) = 1$ if and only if $\mathsf{p}(v) \in A$ and $\mathsf{n}(v) \notin A$
  - $m(v) = 0$ if and only if $\mathsf{p}(v) \notin A$ and $\mathsf{n}(v) \in A$
  - $m(v) = \star$ if and only if $\mathsf{p}(v) \in A$ and $\mathsf{n}(v) \in A$

Counting Formulation for C-FIX-3 and C-MTS-3

1. Capture all FIXs/MTSs ✓

2. Satisfy Phenotype

3. Capture all FIXs/MTSs over perturbations

Satisfy Phenotype (2/3)

A *phenotype* $\beta \equiv \bigwedge_i (v_i \longleftrightarrow e_i)$, where $e_i \in \{0, 1, \star\}$.

Data: Phenotype β
Result: ASP Program Q
Algorithm PhenToASP(β)

```
Q ← ∅
foreach (v ⟷ e) ∈ β do
    if e = 1 then
        Q.add(⊥ ← ∼p(v),    ⊥ ← n(v))
    else if e = 0 then
        Q.add(⊥ ← p(v),    ⊥ ← ∼n(v))
    else if e = ⋆ then
        Q.add(⊥ ← ∼p(v),    ⊥ ← ∼n(v))
end
return Q
```

Satisfy Phenotype (2/3)

A *phenotype* $\beta \equiv \bigwedge_i (v_i \longleftrightarrow e_i)$, where $e_i \in \{0, 1, \star\}$.

Data: Phenotype β
Result: ASP Program Q
Algorithm PhenToASP(β)

```
Q ← ∅
foreach (v ⟷ e) ∈ β do
    if e = 1 then
        Q.add(⊥ ← ∼p(v),    ⊥ ← n(v))
    else if e = 0 then
        Q.add(⊥ ← p(v),    ⊥ ← ∼n(v))
    else if e = ⋆ then
        Q.add(⊥ ← ∼p(v),    ⊥ ← ∼n(v))
end
return Q
```

ASP Encodings

$$\mathcal{P} = \mathrm{PhenToASP}(\beta) \wedge \begin{cases} \mathrm{fASP}(\mathcal{N}) & \text{for FIXs satisfying } \beta \\ \mathrm{tsconj}(\mathcal{N}) & \text{for MTSs satisfying } \beta \end{cases}$$

Counting Formulation for C-FIX-3 and C-MTS-3

1. Capture all MTSs/FIXs ✓

2. Satisfy Phenotype ✓

3. Capture MTSs/FIXs over perturbations

Counting over Perturbations (3/3)

Definition (New BN)

Given a BN $\mathcal{N}$ and a set of perturbable variables $\mathcal{X}$, we construct a new BN $\overline{\mathcal{N}}$ such that for every $v \in \mathrm{Var}(f)$, if $v \in \mathrm{Var}(f) \setminus \mathcal{X}$, then the variable $v \in \mathrm{Var}(\overline{\mathcal{N}})$ and,

$$\overline{f_v} = f_v$$

if $v \in \mathcal{X}$, then three variables $v, v^k, v^o \in \mathrm{Var}(\overline{\mathcal{N}})$ and

$$\begin{aligned}
\overline{f_v} &= \neg v^k \wedge (v^o \vee f_v), \\
\overline{f_{v^k}} &= v^k, \\
\overline{f_{v^o}} &= v^o \wedge \neg v^k.
\end{aligned}$$

Counting over Perturbations (3/3)

Definition (New BN)

Given a BN $\mathcal{N}$ and a set of perturbable variables $\mathcal{X}$, we construct a new BN $\overline{\mathcal{N}}$ such that for every $v \in \mathrm{Var}(f)$, if $v \in \mathrm{Var}(f) \setminus \mathcal{X}$, then the variable $v \in \mathrm{Var}(\overline{\mathcal{N}})$ and,

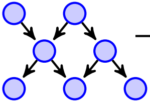
$$\overline{f_v} = f_v$$

if $v \in \mathcal{X}$, then three variables $v, v^k, v^o \in \mathrm{Var}(\overline{\mathcal{N}})$ and

$$\begin{aligned} \overline{f_v} &= \neg v^k \wedge (v^o \vee f_v), \\ \overline{f_{v^k}} &= v^k, \\ \overline{f_{v^o}} &= v^o \wedge \neg v^k. \end{aligned}$$

v^k	v^o	f_v	Interpretation
1	0	0	knockout perturbation
0	1	1	over-expression perturbation
0	0	f_v	v is unperturbed
1	1	-	infeasible due to $\overline{f_{v^o}} = v^o \wedge \neg v^k$

Counting Formulation of 3rd Problems



Given $\mathcal{N}, \beta, \mathcal{X}$

$$\longrightarrow \quad \mathcal{P} = \begin{matrix} \star \leftarrow \star \\ \star \leftarrow \star \\ \star \leftarrow \star \end{matrix} \quad \longrightarrow \quad \#\mathsf{AS}(\mathcal{P})_{\downarrow\Delta}$$

$$\Delta = \{\ldots\}$$

ASP Encodings

$$\mathcal{P} = \mathsf{PhenToASP}(\beta) \wedge \begin{cases} \mathsf{fASP}(\overline{\mathcal{N}}) & \text{for C-FIX-3} \\ \mathsf{tsconj}(\overline{\mathcal{N}}) & \text{for C-MTS-3} \end{cases}, \qquad \overline{\mathcal{N}} \text{ is the new BN}$$

$$\Delta = \bigcup_{v \in \mathcal{X}} \{v^k, v^o\}$$

Experimental Evaluation

Benchmark

Total 645 Boolean Networks from BN literature [TBP+2024,TBS2023]:

- 245 real-world
- 400 randomly generated

with up to 5,000 variables.

Phenotype and Perturbables Variables Selection

- pseudo-randomly fixed three variables to represent the target phenotype
- pseudo-randomly selected up to 50 perturbable variables

Baseline

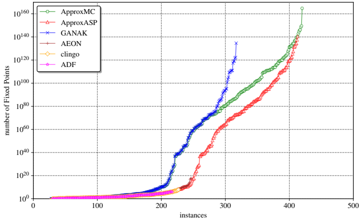
ASP	BDD	ADF	SAT[1]
Clingo ApproxASP	AEON	k++ADF	GANAK ApproxMC

Experimental Settings: 8 GB memory limit and 5000 seconds timeout

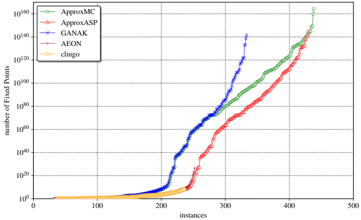
[1] #SAT-based techniques can only be used for fixed points counting.

Results of Counting FIXs

	AEON	ADF	clingo	GANAK	ApproxMC	ApproxASP
C-FIX-1	247	217	227	317	**420**	413
C-FIX-2	252	-	236	333	**438**	429
C-FIX-3	248	-	99	286	600	**645**



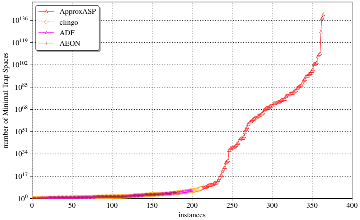
C-FIX-1



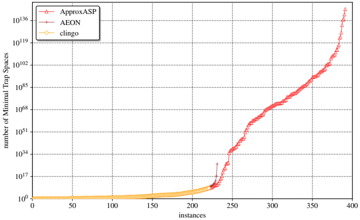
C-FIX-2

Results of Counting MTSs

	AEON	ADF	clingo	ApproxASP
C-MTS-1	179	200	211	**364**
C-MTS-2	231	-	308	**464**
C-MTS-3	148	-	84	**644**



C-MTS-1



C-MTS-2

A Case Study of Interferon 1 model

- Interferon 1: Biochemical species closely tied to immune response present in T-cells.
- The BN model has 121 variables and 55 inputs — not regulated by others.
- The model defines three phenotype variables,
  - ISG (expression antiviral response phenotype)
  - PCK (Proinflammatory cytokine expression inflammation)
  - IFN (Type 1 IFN response)
- We selected 20 variables of the model as potential perturbation targets, which results in 3^{20} admissible perturbations in our BN.
- 🔑 Key idea: robustness $\propto \#\mathrm{AS}(P)_{\downarrow\mathcal{X}}$

A Case Study of Interferon 1 model

- Interferon 1: Biochemical species closely tied to immune response present in T-cells.
- The BN model has 121 variables and 55 inputs — not regulated by others.
- The model defines three phenotype variables,
  - ISG (expression antiviral response phenotype)
  - PCK (Proinflammatory cytokine expression inflammation)
  - IFN (Type 1 IFN response)
- We selected 20 variables of the model as potential perturbation targets, which results in 3^{20} admissible perturbations in our BN.
- 🔑 Key idea: robustness $\propto \#\mathrm{AS}(P)_{\downarrow\mathcal{X}}$

ISG	PCK	IFN	C-MTS-3	Robustness (r)	Robustness
1	-	-	3486784401	1.000	↑
-	1	-	2114072298	0.606	↓
-	-	1	2313362673	0.663	
0	0	0	478296900	0.137	↓
0	1	0	478296900	0.137	↓
1	0	1	1096362783	0.314	
1	1	1	1409735826	0.404	↑

Conclusion

- We address the problem of counting minimal trap spaces and fixed points in BNs.
- We propose novel methods for determining trap space and fixed point counts using approximate model counting, thus entirely avoiding costly enumeration.
- We address three biologically motivated problems:
  - general counting
  - counting occurrences of a known phenotype
  - counting of perturbations that ensure the emergence of a known phenotype
- Approximate counting substantially improves the feasibility of counting in BNs.

Conclusion

- We address the problem of counting minimal trap spaces and fixed points in BNs.
- We propose novel methods for determining trap space and fixed point counts using approximate model counting, thus entirely avoiding costly enumeration.
- We address three biologically motivated problems:
  - general counting
  - counting occurrences of a known phenotype
  - counting of perturbations that ensure the emergence of a known phenotype
- Approximate counting substantially improves the feasibility of counting in BNs.

`https://github.com/meelgroup/bn-counting`