Jin-feng PAN, Min MENG, 2020. Optimal one-bit perturbation in Boolean networks based on cascading aggregation. *Frontiers of Information Technology & Electronic Engineering*, 21(2):294-303. https://doi.org/10.1631/FITEE.1900411

Optimal one-bit perturbation in Boolean networks based on cascading aggregation

Key words: Large-scale Boolean network; Attractor; Cascading aggregation; One-bit perturbation

Corresponding author: Jin-feng PAN
E-mail: panjinfeng1989@163.com
ORCID: https://orcid.org/0000-0001-8567-3121

Motivation

1. Finding all attractors of a network and analyzing the change of the state transition network after intervention play important roles in protecting desired attractors from being damaged.

2. As the simplest structure intervention, one-bit perturbation has attracted much attention in recent years. The main challenge lies in the computational complexity.

3. If the network can be partitioned by cascading aggregation, then the computational complexity to find the attractors of Boolean networks is reduced.

Main idea

1. After one-bit perturbation, desired attractors cannot be destroyed.

2. No new attractors are generated.

3. After one-bit perturbation, if the perturbed states still enter into the original desired attractors or the undesired attractors, this kind of perturbed state is not taken into consideration.

4. The one-bit perturbation occurs on root blocks or blocks with inputs.

Method

1. Find the cascading aggregation that partitions the network into several blocks.

2. Find the one-bit perturbations that destroy the desired attractors.

3. Check the one-bit perturbations that cause the emergence of new attractors.

4. Find the perturbed states entering into different sets of attractors after one-bit perturbation.

Method (Cont'd)

5. For each perturbed states obtained in step 4, calculate Δ_B.

6. Seek out the largest Δ_B, and the corresponding one-bit perturbation is the optimal one-bit perturbation.

Problem formulation

Consider the following BN, which contains n state nodes, $x_1, x_2, \ldots, x_n$:

$$\begin{cases} x_1(t+1) = f_1\left(\left[x_j(t)\right], j \in N_1\right), \\ x_2(t+1) = f_2\left(\left[x_j(t)\right], j \in N_2\right), \\ \qquad\qquad \vdots \\ x_n(t+1) = f_n\left(\left[x_j(t)\right], j \in N_n\right), \end{cases} \tag{1}$$

where $x_i(t)$ can take 1 or 0 and f_i $(i = 1, 2, \ldots, n)$ are logical functions. $N_i \subseteq \{1, 2, \ldots, n\}$ is an index set, which expresses the adjacency relation of nodes corresponding to genes.

Problem formulation (Cont'd)

Desired attractors:

$$A_{\mathrm{d}} = \{d_1, d_2 \ldots, d_\eta\}, \tag{2}$$

Undesired attractors:

$$A_{\mathrm{ud}} = \{\mathrm{ud}_1, \mathrm{ud}_2 \ldots, \mathrm{ud}_\epsilon\}. \tag{3}$$

Objective function:

$$\begin{aligned} \Delta_B = & \sum_{d_i \in A_{\mathrm{d}}} \left[B_j^{(p)}(d_i) - B(d_i) \right] \\ & - \sum_{\mathrm{ud}_o \in A_{\mathrm{ud}}} \left[B_j^{(p)}(\mathrm{ud}_o) - B(\mathrm{ud}_o) \right], \end{aligned} \tag{4}$$

where $B(d_i)$ and $B_j^{(p)}(d_i)$ denote the size of the BOAs for attractor d_i before and after perturbation $f_j^{(p)}$, respectively. This objective function has been proposed in Hu et al. (2016).

Problem formulation (Cont'd)

Definition 1 (Zhao et al., 2016) The aggregation is said to be cascading if (with possible reordering) the groups

$$\chi^i = \cup_{j=q(1)}^{q(i)} \chi_j \ (i = 1, 2, \ldots, s) \tag{6}$$

have zero in-degree, where $q(\cdot)$ is a one-to-one mapping from $\{1, 2 \ldots, s\}$ to itself.

Therefore, the subnetwork Σ_i with nodes χ_i and inputs $\mathcal{U}_i$ is a Boolean control network, which can be described as

$$\begin{aligned} \Sigma_i : x_{ij}(t+1) = & f_{ij} \left(x_{i1}(t), x_{i2}(t), \ldots, x_{in_i}(t), \right. \\ & \left. u_{i1}(t), u_{i2}(t), \ldots, u_{im_i}(t)\right) \end{aligned} \tag{5}$$

for $i = 1, 2, \ldots, s$ and $j = 1, 2, \ldots, n_i$.

Problem formulation (Cont'd)

Example 1

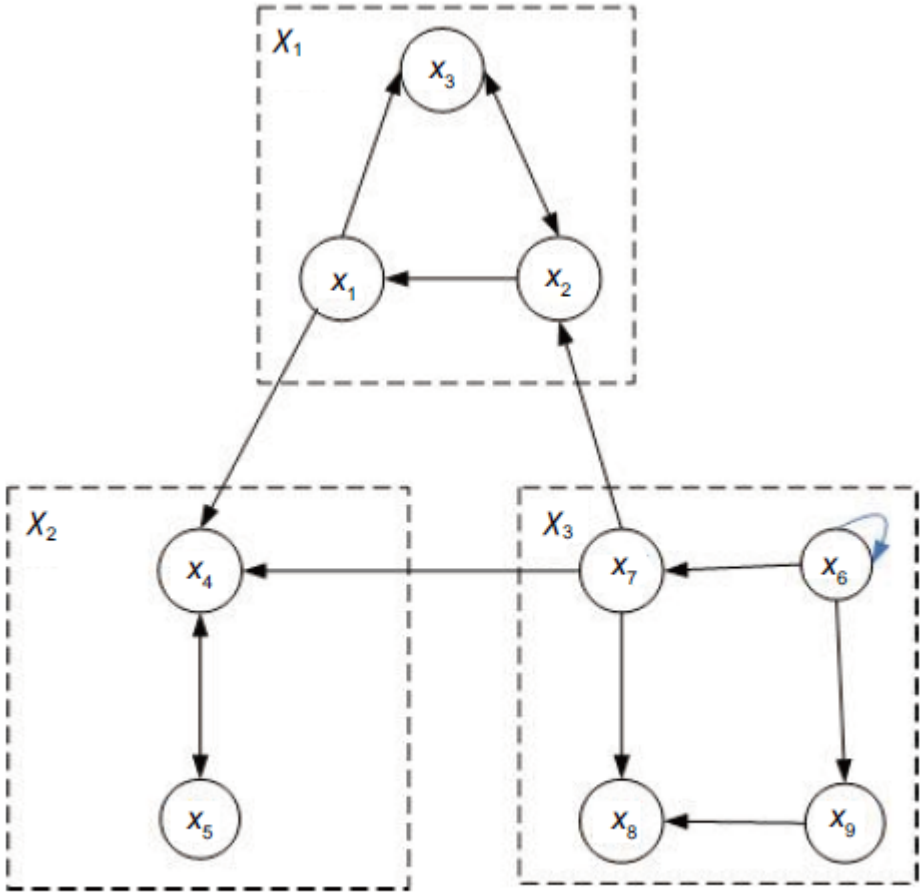

Fig. 1 Aggregation of a network with nine nodes into three Boolean control networks

Major results

Algorithm 1 Seeking ipred(a)

Input: Set $\Omega = \{1, 2, \ldots, s\}$, $V_i = [\,], i \in \Omega$, $c_i = 0$
Output: V^{s^1} // ipred(a)

1: **for all** $k \in \Omega$ **do**
2: $s_k \leftarrow \Psi(a, \chi_k)$
3: **for** $i_1 = 0$ to size$(B_1, 1)$ **do**
4: **if** $B_1(i_1, 2) = s_1$ **then**
5: $V_1[c_1 + +] \leftarrow B_1(i_1, 1)$
6: **end if**
7: **end for**
8: $V_1 \leftarrow [V_{11}, V_{12}, \ldots, V_{1(2^{m_2})}]$ //$\Psi(V_{1j}, \mathcal{U}_2) = j - 1$ // $(j = 1, 2, \ldots, 2^{m_2})$
9: **end for**
10: **for all** $l \in \Omega \backslash \{1\}$ **do**
11: **for** $i_l = 1$ to size$(B_l, 1)$ **do**
12: **for** $r_l = 1$ to 2^{m_l} **do**
13: **if** $B_l(i_l, 2) = s_l$ **and** $B_l(i_l, 3) = r_l - 1$ **then**
14: $V_{lr_1}[c_l + +] \leftarrow B_l(i_l, 1)$
15: **end if**
16: $V_l \leftarrow [V_{l1}, V_{l2}, \ldots, V_{l(2^{m_l})}]$
17: $V^{1^1} \leftarrow V_1$
18: $V^{l^1} \leftarrow V^{(l-1)^1} \bar{\otimes} V_l$
19: **end for**
20: **end for**
21: **end for**

Major results (Cont'd)

Algorithm 2 Identifying new attractors in the root block

Input: Set $B_1 = [\,]$, $C_1 = [\,]$, $R = 0$

Output: R

```
1: for i = 0 to size(A_1, 2) do
2:     for j = 0 to size(Ā, 1) do
3:         B_1(1, i) ← A_1(1, i) and
           B_1(j + 1, i) ← Ā(B_1(j, i) + 1, 2)
4:         for k = 1 to j do
5:             if B_1(j + 1, i) = B_1(k, i) then
6:                 C_1(i) ← B_1(k, i)
7:             end if
8:         end for
9:     end for
10: end for
11: if C_1 ∩ Ã = ∅ then
12:     R ← 0 // No new attractors emerge
13: else
14:     R ← 1 // The one-bit perturbation causes the
          // emergence of new attractors
15: end if
```

Major results (Cont'd)

Algorithm 3 Identifying new attractors in blocks with inputs

Input: Set $B_2 = \text{cell}(1, \xi)$, $R = 0$
Output: R

1: **for** $i = 1$ to size$(\bar{B}, 1)$ **do**
2: **for** $j = 1$ to ξ **do**
3: **for** $l = 2$ to size$(\hat{B}, 1)/2^{m_e-1} - 1$ **do**
4: **if** $\bar{B}(i, 3) \cap C_j(, k_j + 1) = s_{ij}$ **then**
5: $B_2\{1, j\}(1, i) \leftarrow \bar{B}(i, 1)$
6: $B_2\{1, j\}(2, i) \leftarrow \bar{B}(i, 2)$
7: $B_2\{1, j\}(l + 1, i) \leftarrow \hat{B}(B_2\{1, j\}(l, i) \cdot 2^{m_e-1} + C_j(l, k_j + 1) + 1, 2)$
8: **if** there exists p such that $\{B_2\{1, j\}(p \cdot l_i + s_{ij}, i), B_2\{1, j\}(p \cdot l_i + s_{ij} + 1, i), \ldots, B_2\{1, j\}((p+1) \cdot l_i + s_{ij}, i)\} \cap \{C_j(, 1), C_j(, 2), \ldots, C_j(, k_j)\} = \varnothing$ **then**
9: $R \leftarrow 0$ // No new attractors emerge
10: **else**
11: $R \leftarrow 1$ // The one-bit perturbation causes // the emergence of new attractors
12: **end if**
13: **end if**
14: **end for**
15: **end for**
16: **end for**

Major results (Cont'd)

Algorithm 4 Finding the corresponding attractor for a given state a

1: Find the corresponding attractor for state $\Phi(a, \Sigma_1)$. Put the transition states into a row vector D_1, which includes the attractor states
2: For each state in D_1, find the corresponding inputs to Σ_2 and put the corresponding states into a row vector D_{12}
3: D_2 is the set of states in Σ_2 with $D_2(1) = \Psi(a, \chi_2)$, and the $(j+1)^{\text{th}}$ element of D_2 is determined by $D_1(j)$ and $D_{12}(j)$
4: $[D_{12}, D_2]$ does not contain input-state cycles, enlarging D_1 and D_{12} with the multiple of the length of attractor until $[D_1, D_{12}]$ has an input-state cycle
5: Regard system $\Sigma_1 * \Sigma_2$ as the new root block, and construct matrix D_3 via the similar method
6: Construct matrices $D_4, D_5, \ldots, D_s$ in a similar way
7: Choosing the last elements of matrices D_1, D_2, ..., D_s, and arranging them in sequence to form a row vector, the attractor that state a will eventually enter can be found

Major results (Cont'd)

Algorithm 5 Calculating Δ_B after one-bit perturbation

Input: Set $C = [c_1, c_2, \ldots, c_l]$ // the perturbed states
// that enter different attractors after the one-bit
// perturbation; assuming that the maximum
// distance between state c_i and the states that can
// eventually reach it is k_i

Output: Δ_B

1: **for** $i = 1$ to l **do**
2: $C_{i1} \leftarrow \text{ipred}(c_i)/C$
3: **for** $j = 2$ to k_i **do**
4: $C_{ij} = \text{ipred}(C_{i(j-1)})/C$ // C_{ij} is the difference
// between the set of the immediate
// predecessors of each state in $C_{i(j-1)}$ and C
5: **end for**
6: **if** c_i enters a desired attractor before perturbation, and eventually enters an undesired attractor after one-bit perturbation **then**
7: $\Delta_{B_i} \leftarrow -\sum_{t=1}^{k_i} |C_{it}|$
8: **end if**
9: **if** c_i enters an undesired attractor before perturbation, and eventually enters a desired attractor after one-bit perturbation **then**
10: $\Delta_{B_i} \leftarrow \sum_{j=1}^{k_i} |C_{ij}|$
11: **end if**
12: $\Delta_B \leftarrow \sum_{i=1}^{l} \Delta_{B_i}$
13: **end for**

Conclusions

1. The optimal one-bit perturbation of large-scale Boolean networks (BNs) has been carried on the basis that the BN can be partitioned by network aggregation.
2. After aggregation, the problem has been reduced to finding the optimal one-bit perturbation for the corresponding sub-networks.
3. Methods to find the optimal one-bit perturbation have been proposed based on Algorithms 4 and 5.