GAPORE

Liu, Xiang; Wang, Yan; Shi, Ning; Ji, Zhicheng; He, Shan

DOI:
10.1016/j.knosys.2021.107277

Graphical Abstract

GAPORE: Boolean network inference using a genetic algorithm with novel polynomial representation and encoding scheme

Xiang Liu,Yan Wang,Ning Shi,Zhicheng Ji,Shan He

Highlights

GAPORE: Boolean network inference using a genetic algorithm with novel polynomial representation and encoding scheme

Xiang Liu,Yan Wang,Ning Shi,Zhicheng Ji,Shan He

- A novel genetic algorithm incorporating local search is introduced to infer Boolean networks accurately.
- An efficient symbolic polynomial representation is proposed to represent the unknown Boolean functions.
- A novel encoding scheme is developed to encode the Boolean functions flexibly.
- An l_2-norm regularization is designed to reduce the over-fit problem in Boolean network inference.

GAPORE: Boolean network inference using a genetic algorithm with novel polynomial representation and encoding scheme

Xiang Liu[a], Yan Wang[a,*], Ning Shi[b], Zhicheng Ji[a] and Shan He[b]

[a] *School of Internet of Things Engineering, Jiangnan University, Wuxi, 214122, China*
[b] *School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK*

ARTICLE INFO

Keywords:
Data-driven
Boolean network
Polynomial Boolean representation
Hybrid genetic algorithm
Dominant bit encoding scheme

ABSTRACT

Inferring Boolean networks is crucial for modeling and analyzing gene regulatory networks from a systematic perspective. However, the state-of-the-art algorithms cannot accurately infer the topology and dynamics of Boolean networks due to the lack of an efficient approach to representing the unknown Boolean functions and the over-fit problem caused by the noise in time-series data. To address these problems, we propose a novel inference algorithm using a genetic algorithm with novel polynomial representation and encoding scheme (GAPORE) to reconstruct large-scale Boolean networks accurately. First of all, a novel symbolic polynomial representation method is introduced to efficiently represent the unknown Boolean functions of the candidate Boolean network as the symbolic polynomial dynamical equations. Then, a novel encoding scheme is developed to flexibly encode the symbolic polynomial dynamical equations by varying the effective lengths of the chromosomes. To reduce the over-fit problem, the l_2-norm regularization is designed into the fitness evaluation in view of the network sparsity. In addition, the local search strategy is embedded into the hybrid genetic algorithm framework to strengthen the search capability. Extensive experiments demonstrate that GAPORE can infer the large-scale Boolean networks more accurately than state-of-the-art algorithms from the noisy time-series data.

1. Introduction

A gene regulatory network is composed of interactions among the genes and proteins in living cells and reflects the relationship of these genes or proteins [26, 33]. Gene regulatory networks play an increasingly crucial role in uncovering and analyzing the underlying regulatory mechanism of biological organisms from a systematic view [14, 43, 33, 36]. There are many approaches to reconstruct gene regulatory networks by using time-series data from experimental observation, such as fuzzy cognitive maps [37, 38], Bayesian networks[24], Boolean networks [4, 39]. Due to the periodic characteristic and the efficiency to describe signal transduction, Boolean networks have been widely utilized to demonstrate and analyze gene regulatory networks [18].

A growing number of data-driven algorithms have been proposed to infer Boolean functions in the form of logical operators. Among these algorithms, Best-Fit [20] exhaustively searches all the candidate Boolean functions and is vulnerable to the over-fit problem when dealing with the noisy time-series data. MIBNI [3] employs the mutual information theory to reconstruct every Boolean function with a fixed in-degree. Nevertheless, it can only infer a conjunction or disjunction Boolean functions and cannot infer complex Boolean functions. MAGANI [25] utilizes an improved genetic algorithm (GA) to search the candidate Boolean functions. However, the inference performance of these algorithms is far from satisfactory when applied to some complex Boolean functions. One of the main reasons is that Boolean functions in the form of logical operators may be redundant and indefinite [39]. In view of this problem, ATEN [39] introduces a recursive feature reconstruction and elimination (RFRE) procedure to obtain a minimum Boolean function by eliminating the redundant term. Despite these solid achievements, state-of-the-art algorithms could not perform well when they are applied to large-scale Boolean network inference.

To overcome the difficulty of representation, researchers have developed many algebraic expression approaches to representing Boolean functions efficiently. A state space method [6, 7, 30] uses the state-transition matrix based on the semi-tensor product (STP) to express the Boolean functions. However, this approach suffers from the curse of dimensionality since the dimension of the state-transition matrix grows exponentially with the number of nodes [45]. The Zhegalkin polynomial identification method is proposed by [8, 9] to represent and identify the unknown Boolean functions. Nevertheless, there are strict rules for the feasible combinations of all the Zhegalkin coefficients [9], which hinders the spread and application. Jarrah et al. [17] introduce an efficient transformation theory that transforms the logical operators into a polynomial representation. Based on the transformation theory proposed by [17], we design a novel symbolic polynomial representation method to represent the unknown Boolean functions as the symbolic Boolean polynomial dynamical equations.

As analyzed in [40], there are $\left(\frac{2^{2^K} N!}{(N-K)!}\right)^N$ candidate solutions for an unknown Boolean network with N nodes and the maximum in-degree K. Therefore, it is a challenging task to accurately reconstruct the Boolean network from the noisy time-series data. It is well known that genetic algorithms with variable-length chromosomes have an advantage in solving graph optimization problems [34]. Ab-

*Corresponding author
wangyan88@jiangnan.edu.cn (Y. Wang)
ORCID(s): 0000-0002-7700-8719 (Y. Wang)

baspour et al. use the GA with variable-length chromosomes to solve the tour planning problems [2]. Lee designs a GA with variable-length chromosomes for the robot to discover a feasible path without intersecting any obstacles [22]. Li et al. provide a deep insight into the variable-length multiobjective optimization problems [23]. Viswambaran et al. employ a GA with variable-length chromosomes to evolve deep recurrent neural networks (DRNNs) by using the variable-length encoding strategy to represent DRNNs of different depths [42]. The common characteristic of these graphs is that their feasible structures consist of an unfixed or unknown number of nodes or layers. Intrinsically, the Boolean network to be inferred is a directed graph that is suitable to be optimized by a genetic algorithm with variable-length chromosomes. Inspired by this, we design a novel encoding scheme to flexibly encode the Boolean functions of the nodes with different in-degrees.

Moreover, local search (LS) belongs to a category of meta-heuristic algorithms which are very effective on a range of combinational optimization problems [41, 35, 10]. Considering the complexity of the network inference, we incorporate the local search into the genetic algorithm framework to improve the inference performance.

In practice, many inference algorithms suffer from the over-fit problem since the time-series data are noisy, which means that the in-degrees of the nodes inferred by existing algorithms tend to be larger than the real in-degrees of the nodes. A well known method to restrain the over-fit problem is to design a l_2-norm regularization with respect to the regression parameters [11, 5, 27, 19, 44]. Taking the sparsity of networks [40, 43] into account, we design the l_2-norm regularization to the in-degree for the sake of overcoming the over-fit problem. The main contributions of this study can be summarized as follows:

(1) A novel inference algorithm GAPORE is introduced to better infer large-scale Boolean networks. By incorporating the local search strategy into the genetic algorithm, GAPORE is capable to effectively explore and exploit the search space.

(2) To overcome the inefficiency of logical operation representation, a novel symbolic polynomial representation method is proposed to efficiently represent the unknown or undetermined Boolean functions.

(3) Inspired by the variable-length encoding scheme, a novel dominant bit encoding scheme is developed to encode the unknown Boolean functions of the nodes with flexible in-degrees, which enables the flexibility of solutions and maintains the ease of implementation.

(4) By designing the l_2-norm regularization to the in-degree of each node, GAPORE effectively handles the over-fit problem and achieves the best performance in comparison to other algorithms for inferring large-scale Boolean networks.

2. Preliminaries

This section illustrates the related concepts of Boolean networks and the polynomial representation of the deterministic Boolean functions based on the transformation theory proposed by [17, 21].

2.1. Boolean network

A Boolean network $G(V, F)$ is a discrete-time non-linear dynamical system, which consists of a set of nodes $V = \{x_1, x_2, \cdots, x_n, \cdots, x_N\}$ and a set of Boolean functions $F = \{f_1, f_2, \cdots, f_n, \cdots, f_N\}$. In a Boolean network, each node has only a state (0 or 1) at a time point, where 0 represents the state of the node is off (not expressed) and 1 denotes the state of the node is on (expressed). The model of a Boolean network is shown as follows:

$$\begin{cases} x_1(t+1) = f_1\left(x_{1_1}(t), x_{1_2}(t), \cdots, x_{1_{K_1}}(t)\right) \\ \vdots \\ x_n(t+1) = f_n\left(x_{n_1}(t), x_{n_2}(t), \cdots, x_{n_{K_n}}(t)\right) \\ \vdots \\ x_N(t+1) = f_N\left(x_{N_1}(t), x_{N_2}(t), \cdots, x_{N_{K_N}}(t)\right) \end{cases} \tag{1}$$

where f_n denotes a Boolean function. $x_n(t+1)$ is the state of the target node x_n at time point $t+1$. $x_{n_{K_n}}(t)$ is the state of the regulatory node $x_{n_{K_n}}$ at time point t. In addition, K_n is the in-degree of the node x_n, denoting the number of regulatory nodes of the target node x_n. We define $K = max\{K_n\}$ $(n = 1, \cdots, N)$ as the maximum in-degree of the Boolean network.

The state of the Boolean network at time point t is represented as $\boldsymbol{x}^{\mathrm{T}}(t) = [x_1(t), x_2(t), \cdots, x_N(t)]$. The Boolean network evolves dynamically from one state $\boldsymbol{x}(t)$ to the next state $\boldsymbol{x}(t+1)$, governed by the set of Boolean functions. With the advances in high-throughput microarray techniques, more time-series data can be observed and denoted as $\boldsymbol{D} = [\boldsymbol{x}(1), \cdots, \boldsymbol{x}(\rho)] \in \mathbb{R}^{n\times\rho}$, where ρ is the number of time points. In addition, the Boolean network has a periodic characteristic called attractor, which corresponds to functional cellular states. Intrinsically, the attractors are the steady states of the Boolean network, and the concept of attractors is defined [1] as follows.

Definition 1. *For the Boolean network (1), the sequence of states $\boldsymbol{x}(t), \boldsymbol{x}(t+1), \cdots, \boldsymbol{x}(t+T-1)$ is defined as a periodic attractor if they satisfy $\boldsymbol{x}(t+T) = \boldsymbol{x}(t)$, $\boldsymbol{x}(t+i) \neq \boldsymbol{x}(t)$, $i \in [1, T-1]$. The period of the attractor is T.*

As shown in Fig. 1, we take a small Boolean network with three nodes as a simple example to intuitively illustrate the related concepts mentioned above, including the available time-series data, the network topology, the state transitions or dynamics, and the set of Boolean functions. In addition, the Boolean network has an attractor ($101 \rightarrow 101$) with $T = 1$ and an attractor ($110 \rightarrow 100 \rightarrow 001 \rightarrow 110$) with $T = 3$.

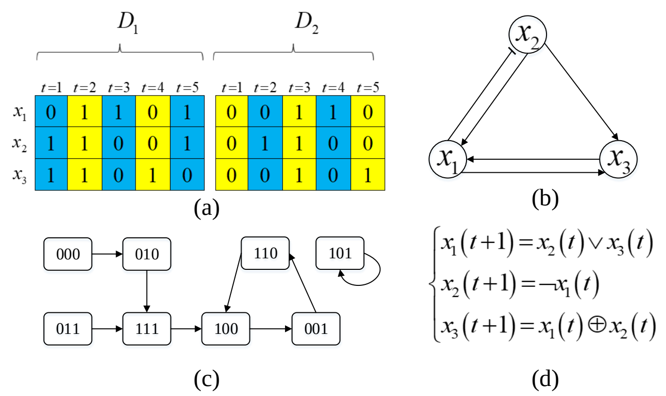

D_1
D_2
$t=1$ $t=2$ $t=3$ $t=4$ $t=5$
x_1 0 1 1 0 1 | 0 0 1 1 0
x_2 1 1 0 0 1 | 0 1 1 0 0
x_3 1 1 0 1 0 | 0 0 1 0 1
(a)
x_1
x_2
x_3
(b)
000
010
110
101
011
111
100
001
(c)
$x_1(t+1)=x_2(t)\vee x_3(t)$
$x_2(t+1)=\neg x_1(t)$
$x_3(t+1)=x_1(t)\oplus x_2(t)$
(d)

Figure 1: A simple example for a Boolean network with three nodes. In (a), there are two time-series D_1, D_2. (b) displays the network topology. (c) demonstrates the state transitions. (d) shows the Boolean functions using logical operators, where operator $\oplus$ denotes the logical operator *XOR*.

2.2. Data-driven Boolean network inference

The wide application of high-throughput microarray techniques facilitates time-series measurements of genes, which provides adequate time-series data for Boolean network reconstruction. The data-driven inference is to find the candidate Boolean network whose state transitions are the most consistent with the time-series data. The Boolean network inferred from the time-series data is assumed as follows:

$$\begin{cases} \hat{x}_1(t+1) = \hat{f}_1\left(\hat{x}_{1_1}(t), \hat{x}_{1_2}(t), \cdots, \hat{x}_{1_{K_1}}(t)\right) \\ \vdots \\ \hat{x}_n(t+1) = \hat{f}_n\left(\hat{x}_{n_1}(t), \hat{x}_{n_2}(t), \cdots, \hat{x}_{n_{K_n}}(t)\right) \\ \vdots \\ \hat{x}_N(t+1) = \hat{f}_N\left(\hat{x}_{N_1}(t), \hat{x}_{N_2}(t), \cdots, \hat{x}_{N_{K_N}}(t)\right) \end{cases} \tag{2}$$

where $\hat{x}_n(t+1)$ corresponds to the inferred state of the target node x_n at time point $t+1$ and $\hat{x}_{n_1}(t), \hat{x}_{n_2}(t), \cdots, \hat{x}_{n_{K_n}}(t)$ are the inferred states of the regulators at time point t. $\hat{f}_n$ is the inferred Boolean function of x_n.

As analyzed in [40], each node x_n has $\frac{N!}{(N-K)!}$ possible ordered regulators, and the number of possible networks is $\left(\frac{2^{2^K} N!}{(N-K)!}\right)^N$ in terms of an unknown Boolean network with N nodes and the maximum in-degree K. Thus, it is challenging to infer large-scale Boolean networks accurately from the time-series data.

2.3. The transformation for logical operators

It is inefficient to use logical operators to represent the Boolean functions because the logical expression tends to be redundant and hard to be parameterized [7, 39]. For instance, the Boolean function $y = (\neg a \wedge b \wedge c) \vee (\neg a \wedge b \wedge \neg c) \vee (a \wedge \neg b \wedge \neg c) \vee (a \wedge \neg b \wedge c)$ is equivalent to $y = (\neg a \wedge b) \vee (a \wedge \neg b)$, which implies the variable c is redundant. Moreover, it is difficult to deal with the logical operation because the logical operators have different precedence. Thus, it is necessary to transform the logical relationships into the equivalent algebraic form to avoid the redundancy of logical operation representation. The transformation theory is proposed by [17, 21] to represent the deterministic Boolean function as a specific Boolean polynomial dynamical equation. Specifically, the transformation theory includes two steps as follows:

I. Replace the logical operations with the combination of addition and multiplication according to Eq.(3).

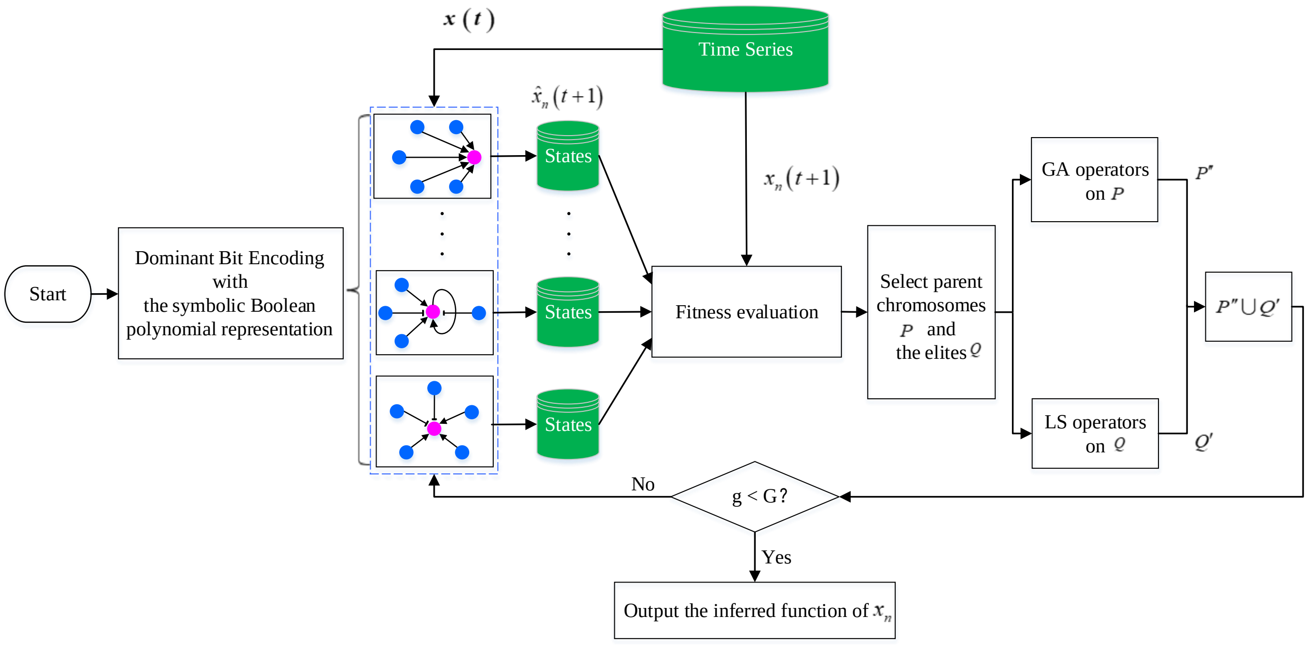
$$\begin{cases} x \wedge y := xy \\ x \vee y := x + y + xy \\ \neg x := x + 1 \end{cases} \tag{3}$$

II. Divide the obtained expression by 2 and then take the remainder.

where x and y are the Boolean variables, the symbols $\wedge$, $\vee$, and $\neg$ represent the logical operators *AND*, *OR*, and *NOT* respectively. In this way, the logical operation expression can be equivalently transformed into a Boolean polynomial expression. It is worth noting that all coefficients of the monomials are binary because of Step II. For simplicity, we omit Step II in the following formula.

3. The proposed inference algorithm

To accurately infer the Boolean network, we propose a novel hybrid genetic algorithm called GAPORE, which combines the improved genetic algorithm and the local search strategy. When the maximum in-degree K is determined, the search space of a Boolean network with N nodes is approximately proportional to $O(N^{KN})$ and is computationally prohibitive to be computed by brute force search [40]. Thus, it is necessary to adopt the decomposition strategy for inferring large-scale Boolean networks. Based on the divide and conquer strategy, GAPORE separately infers the Boolean function of each target node x_n and finally combines the Boolean functions of all target nodes as the Boolean network. Fig. 2 gives an overview of inferring the Boolean function of a target node x_n. For each target node x_n, we first propose the dominant bit encoding scheme (DBES) based on the symbolic polynomial representation to initialize the population. Next, we evaluate each chromosome of the population by a fitness function. Then, we select the chromosomes with better fitness values as two parts: the parent chromosomes P and the elite group Q. We conduct the GA operators on P to generate offspring and the LS operators on Q to exploit the neighborhoods of these elites. Repeat this process until the maximum number of iterations is reached and output the best inferred Boolean function of x_n. Finally, the Boolean network is obtained by combining all the N Boolean functions. The complete framework for inferring the Boolean functions of all nodes in the Boolean network is summarized as Algorithm 1.



x(t)
Time Series
$\hat{x}_n(t+1)$
States
States
States
$x_n(t+1)$
Start
Dominant Bit Encoding with the symbolic Boolean polynomial representation
Fitness evaluation
Select parent chromosomes P and the elites Q
GA operators on P
LS operators on Q
P'
Q'
P''∪Q'
g < G?
No
Yes
Output the inferred function of x_n

Figure 2: Overview of inferring the Boolean function of a target node x_n by the proposed algorithm.

3.1. A novel symbolic polynomial representation

Based on the transformation theory proposed by [17, 21], a deterministic Boolean function can be represented in the form of the Boolean polynomial dynamical equation. However, there is still lack of an effective approach to representing an unknown or undetermined Boolean function in the algebraic form. Therefore, it is significant to design a symbolic polynomial representation approach for the unknown Boolean function.

Consider a target node x_n in an unknown or undetermined Boolean network with N nodes and the maximum in-degree K with the regulatory nodes $\left\{x_{n_1}, \cdots, x_{n_K}\right\}$. Because the in-degree K_n of each target node varies and the coefficients of the Boolean polynomial dynamical system are unknown, it is necessary to design a uniform symbolic expression to deal with these uncertainties.

Obviously, any Boolean polynomial dynamical equation $x_n(t+1) = f\left(x_{n_1}(t), x_{n_2}(t), \cdots, x_{n_K}(t)\right)$ consists of at most 2^K monomials and each monomial can be represented by a basic monomial $x_{n_1}^{b_1} x_{n_2}^{b_2} \cdots x_{n_K}^{b_K}$ where the binary b_i is the power of x_{n_i} $(i \in \{1, \cdots, K\})$. For instance, if $K = 5$, a monomial $x_{n_1}(t)x_{n_2}(t)x_{n_5}(t)$ can be represented using the basic monomial $x_{n_1}^{b_1}(t)x_{n_2}^{b_2}(t)x_{n_3}^{b_3}(t)x_{n_4}^{b_4}(t)x_{n_5}^{b_5}(t)$ by letting $b_1 = 1, b_2 = 1, b_3 = 0, b_4 = 0, b_5 = 1$, respectively.

Considering that a Boolean polynomial dynamical equation consists of at most 2^K monomials, it is significant to represent which monomials appear in the Boolean polynomial dynamical equation. Intuitively, an effective method is to assign a unique binary coefficient to each monomial. If the binary coefficient is 1, the corresponding monomial appears in the equation; otherwise, if the binary coefficient is 0, the corresponding monomial does not appear in the equation. Thus, we design a binary matrix to assign a unique binary coefficient a_i to each monomial (determined by a set of $b_K, b_{K-1}, \cdots, b_1$) as follows:

$$\Psi = \begin{array}{c} \\ a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ \vdots \\ a_{2^K-2} \\ a_{2^K-1} \end{array} \begin{array}{c} \begin{array}{ccccccc} b_K & b_{(K-1)} & \cdots & b_4 & b_3 & b_2 & b_1 \end{array} \\ \left[\begin{array}{ccccccc} 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 1 & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 & 1 & 1 \end{array}\right] \end{array} \quad (4)$$

where K is the maximum in-degree of the Boolean network. In this way, we establish the one-to-one correspondence between the binary coefficients and the monomials. Hence, the Boolean polynomial dynamical equation can be described by the monomials and their coefficients as follows:

$$\begin{aligned} & x_n(t+1) = f\left(x_{n_1}(t), x_{n_2}(t), \cdots, x_{n_K}(t)\right) \\ & = a_0 + a_1 x_{n_1}(t) + a_2 x_{n_2}(t) + a_3 x_{n_1}(t) x_{n_2}(t) + a_4 x_{n_3}(t) \\ & + a_5 x_{n_1}(t) x_{n_3}(t) + a_6 x_{n_2}(t) x_{n_3}(t) + a_7 x_{n_1}(t) x_{n_2}(t) x_{n_3}(t) \\ & + \cdots + a_{2^K-1} x_{n_1}(t) x_{n_2}(t) \cdots x_{n_K}(t) \\ & = \sum_{i=0}^{2^K-1} a_i x_{n_1}^{b_1}(t) x_{n_2}^{b_2}(t) \cdots x_{n_K}^{b_K}(t) \end{aligned} \quad (5)$$

where $\left\{a_0, a_1, \cdots, a_{2^K-1}\right\}$ are all the binary coefficients, and $x_{n_1}^{b_1}(t)x_{n_2}^{b_2}(t) \cdots x_{n_K}^{b_K}(t)$ is the basic monomial for the symbolic Boolean polynomial dynamical equation. And the subscript

i of a_i is a decimal number converted from the corresponding binary number $(b_K, b_{K-1}, \cdots, b_1)$ in the binary matrix Ψ, i.e., $(i)_{10} = (b_K, b_{K-1}, \cdots, b_1)_2$. For instance, the corresponding $(b_K, b_{K-1}, \cdots, b_1)$ of the 7_{th} row in the binary matrix Ψ is $(0, 0, \cdots 0, 1, 1, 1)$. According to Eq.(5), the corresponding monomial is $x^1_{n_1}(t)x^1_{n_2}(t)x^1_{n_3}(t)x^0_{n_4}(t) \cdots x^0_{n_K}(t)$ and a_7 will be assigned to the monomial $x_{n_1}(t)x_{n_2}(t)x_{n_3}(t)$ as the unique binary coefficient.

Remark 1. *The symbolic polynomial representation method provides a novel algebraic tool to efficiently represent the unknown Boolean functions. From Eq.(5), there are 2^K unknown coefficients a_i.Since that each coefficient is binary and the in-degree is K, there exists 2^{2^K} possible Boolean rules. Meanwhile, each node has $\frac{N!}{(N-K)!}$ candidate ordered regulatory nodes. Consequently, the number of the Boolean polynomial dynamical equations is $\frac{2^{2^K} N!}{(N-K)!}$ that coincides with the number of the possible Boolean functions in the form of logical operators [40]. In other words, this symbolic polynomial representation method can provide a complete solution space of Boolean functions, which is different from other algorithms such as MIBNI [3].*

3.1.1. Boolean rules

Interestingly, the symbolic polynomial representation approach has a series of definite coefficient combinations to describe the Boolean rules, corresponding to the logical operators. In this subsection, we deduce the coefficient combinations corresponding to the logical operators such as *OR*, *AND*, *XOR*.

Theorem 1. *Suppose that $x_n(t+1) = x_{n_1}(t) \oplus x_{n_2}(t)$ is a Boolean function with the Boolean rule XOR. Based on Eq.(3), it can be transformed into a symbolic polynomial dynamical equation as follows:*

$$x_n(t+1) = x_{n_1}(t) + x_{n_2}(t) \tag{6}$$

Therefore, the coefficients of the polynomial equation satisfy $a_0 = 0$, $a_1 = 1$, $a_2 = 1$, $a_i = 0$ $(i \in \{3, 4, \cdots, 2^K - 1\})$.

Proof 1. *Considering the Idempotent Law, the Boolean variable x satisfies $x^2 = x$. Since that any even number can be divisible by 2, the even monomial in the equation can be omitted according to Step II of the transformation theory.*

$$\begin{aligned}
x_n(t+1) &= x_{n_1}(t) \oplus x_{n_2}(t) \\
&= (\neg x_{n_1}(t) \wedge x_{n_2}(t)) \vee (x_{n_1}(t) \wedge \neg x_{n_2}(t)) \\
&= (1 + x_{n_1}(t))x_{n_2}(t) + (x_{n_1}(t)(1 + x_{n_2}(t))) \\
&+ (1 + x_{n_1}(t))x_{n_2}(t) \cdot x_{n_1}(t)(1 + x_{n_2}(t)) \\
&= x_{n_1}(t) + x_{n_2}(t) + 2x_{n_1}(t)x_{n_2}(t) + x_{n_1}(t)x_{n_2}(t) \\
&+ x^2_{n_1}(t)x_{n_2}(t) + x_{n_1}(t)x^2_{n_2}(t) + x^2_{n_1}(t)x^2_{n_2}(t) \\
&= x_{n_1}(t) + x_{n_2}(t) + 6x_{n_1}(t)x_{n_2}(t) \\
&= x_{n_1}(t) + x_{n_2}(t)
\end{aligned}$$

The proof is complete.

Similarly, the other definite coefficient combinations of the Boolean rules can be obtained by transforming the corresponding logical operators according to Eq.(3) of the transformation theory [17]. If the Boolean network has the maximum in-degree $K = 5$, then the coefficient combination $A_i = (a_0, a_1, a_2, a_3, \cdots, a_{28}, a_{29}, a_{30}, a_{31})$ can be deduced to represent a concrete Boolean rule. Empirically, ten common Boolean rules $A_1, A_2, \cdots, A_{10}$ are deduced. As summarized in Eq.(7), the coefficient combination A_1 corresponds to the logical operator *NOT*, A_2 is equivalent to the logical operator *NAND*, A_3 corresponds to the logical operator *OR*, A_4 reflects the logical operator *XOR*, and A_5 corresponds to the logical operator *XNOR*. In terms of A_6, it describes an extended *XOR*, which can be expressed as: $x_n(t+1) = [x_{n_1}(t) \wedge \neg x_{n_2}(t)] \vee [\neg x_{n_1}(t) \wedge x_{n_2}(t) \wedge x_{n_3}(t)]$. With respect to A_7, it describes a Boolean rule that $x_n(t+1) = [\neg x_{n_1}(t) \wedge \neg x_{n_2}(t) \wedge \neg x_{n_3}(t)]$. With respect to A_8, it describes a Boolean rules that $x_n(t+1) = x_{n_1}(t) \wedge [x_{n_2}(t) \vee x_{n_3}(t)]$. The coefficient combination A_9 describes a Boolean rules that $x_n(t+1) = x_{n_1}(t) \wedge x_{n_2}(t) \wedge [x_{n_3}(t) \vee x_{n_4}(t)]$. With respect to A_{10}, it describes a Boolean rules that $x_n(t+1) = \neg x_{n_1}(t) \wedge x_{n_2}(t) \wedge [x_{n_3}(t) \vee x_{n_4}(t)]$.

$$\begin{cases}
A_1 = (1100\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1) \\
A_2 = (1010\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 2) \\
A_3 = (0111\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 2) \\
A_4 = (0110\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 2) \\
A_5 = (1110\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 2) \\
A_6 = (0101\ 0011\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 3) \\
A_7 = (1111\ 1111\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 3) \\
A_8 = (0001\ 0101\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 3) \\
A_9 = (0000\ 0001\ 0001\ 0001\ 0000\ 0000\ 0000\ 0000\ 4) \\
A_{10} = (0000\ 0011\ 0011\ 0011\ 0000\ 0000\ 0000\ 0000\ 4)
\end{cases} \tag{7}$$

Considering there are various logical operators with different precedence, it is difficult to encode a number of complex Boolean rules in the form of logical operators. In contrast, the symbolic polynomial representation approach can efficiently describe any complex Boolean rules by using different coefficient combinations which is easier to be encoded. Hence, it is efficient to design a novel encoding scheme based on the symbolic Boolean polynomial representation.

3.2. Dominant Bit Encoding Scheme

Inspired by the variable-length encoding scheme applied in the graph optimization problems with unknown structures [34, 13, 16], we attempt to design a novel flexible encoding scheme for the candidate Boolean functions with different in-degrees. Meanwhile, we are enlightened by the simplicity and ease of implementation of GAs with fixed-length chromosomes. Consequently, we introduce a dominant bit encoding scheme by designing the in-degree K_n as the dominant bit to vary the effective length of the chromosomes and

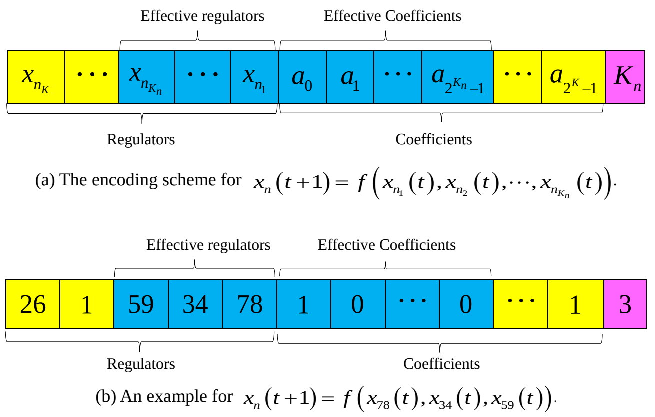

Effective regulators
Effective Coefficients
Regulators
Coefficients
(a) The encoding scheme for $x_n(t+1) = f(x_{n_1}(t), x_{n_2}(t), \cdots, x_{n_{K_n}}(t))$.
(b) An example for $x_n(t+1) = f(x_{78}(t), x_{34}(t), x_{59}(t))$.

Figure 3: The dominant bit encoding scheme. The in-degree K_n of a target node x_n is designed as the dominant bit. (b) is an example of the encoding for the candidate function of x_n in a BN with $N = 100, K = 5, K_n = 3$. Due to the dominant bit $K_n = 3$, the number of effective regulators is 3 ($x_{n_1} = 78, x_{n_2} = 34, x_{n_3} = 59$), and the number of effective coefficients is 8 bits.

keeping the total lengths of chromosomes the same.

As seen from Fig.3, a chromosome is composed of three fragments, including a set of regulators $\Omega = \left\{x_{n_K}, \cdots, x_{n_1}\right\}$, a set of coefficients $\Phi = \left\{a_0, a_1, \cdots, a_{2^K-1}\right\}$ and the in-degree K_n. The effective length of each chromosome is flexible and determined by the dominant bit K_n, shown as the two blue parts $\omega = \left\{x_{n_{K_n}}, \cdots, x_{n_1}\right\}$ and $\phi = \left\{a_0, a_1, \cdots, a_{2^{K_n}-1}\right\}$ in Fig.3 (a). The effective length of the chromosome is $dim = K_n + 2^{K_n} + 1$, which tends to be smaller than the dimension of the search space $Dim = K + 2^K + 1$ due to the network sparsity. Moreover, the size of the search space is $\frac{2^{2^K} N!}{(N-K)!}$ according to the principle of permutation and combination. Based on the symbolic polynomial representation, the dominant bit encoding scheme allows the flexibility of solutions and maintains the ease of implementation by varying the effective length instead of the total length of the chromosome.

3.3. Fitness Evaluation

To evaluate the fitness of the chromosomes, we propose a fitness function with the l_2-norm regularization to the in-degree K_n. Firstly, we decode the chromosome as a candidate Boolean function by identifying the corresponding regulators and coefficients according to their positions in the chromosome. Then, we infer the state transitions by using the candidate Boolean function with the time-series data. Finally, we compare the inferred state transitions and the corresponding values in the time-series data. Taking the network sparsity into consideration, we modify the fitness function by adding the l_2-norm regularization of the in-degree K_n for the sake of handling the over-fit problem.

3.3.1. Inconsistency Evaluation

It is necessary to evaluate the inferred Boolean function of a target node. To this end, the inconsistency should be defined to quantify the difference between the inferred state transitions $\hat{x}_n(t)$ ($t \in \{2, \cdots, \rho\}$) of the target node x_n and the time-series data. Intuitively, if the inferred Boolean function is correct, then the state transitions of the target node will fit the time-series data, and the difference between them tends to be zero. On this basis, the inconsistency evaluation is designed as follows:

$$E_n^j = \sum_{t=2}^{\rho} \left(\hat{x}_n(t) - x_n(t)\right)^2 \tag{8}$$

where ρ is the length (i.e., the number of time points) of the time-series data D_j. $\hat{x}_n(t)$ and $x_n(t)$ are the inferred state and the true value in the time-series data at time point t, respectively.

3.3.2. l_2-norm Regularization

Since the time series data are noisy, many inference algorithms are susceptible to the over-fit problem that the inferred in-degree is larger than the real in-degree. According to related literature [15, 11, 5, 27, 19, 44], one of the popular method to avoid the over-fit problem is l_2-norm regularization. As analyzed in [40], the probability that a node has the in-degree decreases as the value of the in-degree increases, following a power-law distribution. Specifically, a large number of nodes have only a few regulatory nodes. In other words, the Boolean network is sparse network. Taking the sparsity of networks into account, we modify the fitness function by designing a l_2-norm regularization term to the in-degree K_n as follows:

$$E_f = \frac{1}{d \times \rho} \sum_{i=1}^{d} \sum_{t=2}^{\rho} \left(\hat{x}_n(t) - x_n(t)\right)^2 + \frac{\lambda}{2 \times (d \times \rho)} (K_n)^2 \tag{9}$$

where d is the number of time-series D_i and each D_i contains ρ time points.

3.4. Selection

The selection mechanism aims to choose the parent chromosomes with good fitness values from the current population for later variation to generate offspring. We adopt the binary tournament selection strategy owing to its high efficiency and easy implementation. Each time, two chromosomes are randomly selected from the population for competition and the fittest one wins and is selected for variation. We repeat this process until the population size is reached.

3.5. Crossover

A single-point crossover is employed in the proposed algorithm. Firstly, we randomly generate a number r ranging from 0 to 1. The crossover probability is preset as pc. If $r < pc$, we utilize the single-point crossover on the two parents. The crossover point c is chosen randomly. Suppose there are two chromosomes P_i and P_j, where the i_{th} chromosome $P_i = [v_{i,1}, \cdots, v_{i,c}, \cdots, v_{i,K+2^K+1}]$, the j_{th} chromosome $P_j = [v_{j,1}, \cdots, v_{j,c}, \cdots, v_{j,K+2^K+1}]$. The two offspring P_i' and P_j' can be obtained as follows:

$$\begin{aligned} P_i' &= [v_{i,1}, \cdots, v_{i,c}, v_{j,c+1}, \cdots, v_{j,K+2^K+1}], \\ P_j' &= [v_{j,1}, \cdots, v_{j,c}, v_{i,c+1}, \cdots, v_{i,K+2^K+1}]. \end{aligned} \tag{10}$$

3.6. Mutation

The mutation operator is executed with the probability pm on each chromosome $P_i = [v_{i,1}, \cdots, v_{i,j}, \cdots, v_{i,K+2^K+1}]$. As shown in Fig. 3, the chromosome consists of three fragments: the regulators, the binary coefficients, and the in-degree. The first step is to determine the mutation point by randomly generate an integer $j \in [1, K + 2^K + 1]$. The second step is to randomly generate a new value for the variable $v_{i,j}$ to be mutated. Since that the variables in different fragments have different range of values, a new variable $v'_{i,j}$ is generated as follows:

$$v'_{i,j} = \begin{cases} randi(1, N), & 1 < j < K, \\ 1 - v_{i,j}, & K + 1 \leqslant j \leqslant K + 2^K, \\ randi(1, K), & j = K + 2^K + 1. \end{cases} \quad (11)$$

where $randi(1, N)$ is the function to randomly generate an integer ranging from 1 to N. Finally, a new chromosome $P'_i = [v_{i,1}, \cdots, v'_{i,j}, \cdots, v_{i,K+2^K+1}]$ is obtained by Eq.(11).

3.7. Local search with Boolean rules

Local search has proven to be a successful tool for various graph optimization problems [10, 41, 35]. The underlying idea is that better chromosomes can be found within the neighborhood of the current chromosome.

Considering a chromosome Q_i. The neighborhood $H(Q_i)$ is the set of all chromosomes that can be obtained by conducting a given kind of modification on Q_i. In this paper, the modification is specified by the local search operators including the swap operator L_s, the mutation operator L_m, and the replacement operator L_r. The swap operator tries to select two regulators and exchange their position in the fragment $\Omega = \left\{ x_{n_K}, \cdots, x_{n_1} \right\}$. We observe that x_{n_1} is the most possible regulators during the evolution. As a result, the swapped regulators are only selected from $\left\{ x_{n_K}, \cdots, x_{n_2} \right\}$ without x_{n_1} for the purpose of reducing the burden of computation. For instance, if a chromosome is shown as $Q_i = [x_{n_5},\ x_{n_4},\ x_{n_3},\ x_{n_2},\ x_{n_1},\ a_0, \cdots, a_{31}, K_{n_i}]$, the neighborhood obtained by the swap operator can be represented as

$$L_s = \begin{cases} [x_{n_5},\ x_{n_4},\ x_{n_2},\ x_{n_3},\ x_{n_1},\ a_0, \cdots, a_{31}, K_{n_i}], \\ [x_{n_5},\ x_{n_2},\ x_{n_3},\ x_{n_4},\ x_{n_1},\ a_0, \cdots, a_{31}, K_{n_i}], \\ [x_{n_5},\ x_{n_3},\ x_{n_4},\ x_{n_2}\ , x_{n_1},\ a_0, \cdots, a_{31}, K_{n_i}], \\ [x_{n_4},\ x_{n_5},\ x_{n_3},\ x_{n_2}\ , x_{n_1},\ a_0, \cdots, a_{31}, K_{n_i}], \\ [x_{n_3},\ x_{n_4},\ x_{n_5},\ x_{n_2}\ , x_{n_1},\ a_0, \cdots, a_{31}, K_{n_i}], \\ [x_{n_2},\ x_{n_4},\ x_{n_3},\ x_{n_5},\ x_{n_1},\ a_0, \cdots, a_{31}, K_{n_i}]. \end{cases} \quad (12)$$

The mutation operator aims to mutate one of the regulators x_{n_j} $(j \in [1, 5])$ by letting $x'_{n_j} = randi(1, x_{n_j})$ where $randi(1, x_{n_j})$ is the function to randomly generate an integer ranging from 1 to x_{n_j}. The neighborhood obtained by the mutation operator includes

$$L_m = \begin{cases} [x_{n_5},\ x_{n_4},\ x_{n_3},\ x'_{n_2},\ x_{n_1},\ a_0, \cdots, a_{31}, K_{n_i}], \\ [x_{n_5},\ x_{n_4},\ x'_{n_3},\ x_{n_2},\ x_{n_1},\ a_0, \cdots, a_{31}, K_{n_i}], \\ [x_{n_5},\ x'_{n_4},\ x_{n_3},\ x_{n_2},\ x_{n_1},\ a_0, \cdots, a_{31}, K_{n_i}], \\ [x'_{n_5},\ x_{n_4},\ x_{n_3},\ x_{n_2}\ , x_{n_1},\ a_0, \cdots, a_{31}, K_{n_i}]. \end{cases} \quad (13)$$

As mentioned above, there are a set of Boolean rules summarized as the set of coefficient combinations in Eq. (7). To make use of the coefficients corresponding to the certain Boolean rules, we employ the replacement operator L_r to replace the coefficient fragment $\phi = \left\{ a_0, a_1, \cdots, a_{2^{K_n}-1} \right\}$ and the in-degree K_n of the chromosome Q_i with the coefficient combinations A_b $(b = 1, \cdots, 10)$.

With respect to the chromosome Q_i, We change the regulatory nodes by L_s and L_m and replace the coefficient fragment by L_r to generate a set of neighborhoods $H(Q_i)$. Then, we evaluate the fitness of each neighbor chromosome by Eq.(9). Finally, the current solution Q_i is updated by the best neighbor chromosome in $H(Q_i)$.

4. Experiments and Results

To validate the inference performance, we utilize ATEN, Best-Fit, MIBNI, and GAPORE to infer both the topology and the dynamics of ten benchmark networks, including five artificial Boolean networks and five real-world gene regulatory networks. For each experiment, ATEN and GAPORE take the average over 10 runs since that both of them are heuristic algorithms.

It is necessary to employ related measures to evaluate the performance of these algorithms on inferring both the topology and the dynamics. In the context of Boolean network, if a target node x_n has a regulatory node x_{n_j}, then there exists an interaction (or edge) starting from the regulatory node x_{n_j} to the target node x_n. The topology of the Boolean network consists of all the interactions from every regulatory nodes to the target nodes. In terms of the performance on inferring the network topology, we adopt a set of classic metrics [28], namely *Accuracy*, *Recall*, *Precision*, *FPR*, as well as *F-score*. The topology metrics is provided as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (14)$$

$$Recall = \frac{TP}{TP + FN} \quad (15)$$

$$Precision = \frac{TP}{TP + FP} \quad (16)$$

$$FPR = \frac{FP}{FP + TN} \quad (17)$$

Algorithm 1: The pseudo-code of GAPORE

```
input : The time-series data $D_j$ containing $\rho$ time points, $j = 1, 2, \cdots, d$,
        the number of nodes $N$, the maximum in-degree $K$.
output: The inferred Boolean network
1  Preset the maximum iteration $G$, the population size $S$, the elite size $E$, the number of Boolean rules $B$ ;
2  for $n \leftarrow 1$ to $N$ do
3      $P \leftarrow$ Initialize population using the dominant bit encoding scheme with the symbolic Boolean polynomial
       representation;
4      foreach chromosome $P_i$ in $P$ do
5          Decode the chromosome as a Boolean function;
6          Obtain the state transitions of $x_n$ by inputting the time-series data into the Boolean function;
7          Evaluate the fitness of $P_i$ using Eq. (9) ;
8      end
9      g=0;
10     while $g < G$ do
11         $g = g + 1$ ;
12         $Q \leftarrow$ select the best $E$ chromosomes from $P$ according to the fitness as an elite group;
13         $P' \leftarrow$ select $S$ chromosomes from $P$ as parents by tournament selection;
14         $P' \leftarrow$ crossover on each two parent chromosomes in $P'$ by Eq.(10);
15         $P'' \leftarrow$ mutate each chromosome in $P'$ by Eq.(11) ;
16         foreach chromosome $Q_i$ in $Q$ do
17             $H(Q_i) \leftarrow$ Generate the neighborhoods using the local search operators $L_s$, $L_m$, and $L_r$ on $Q_i$ ;
18             foreach chromosome $Q_i$ in $H(Q_i)$ do
19                 Decode the chromosome as a Boolean function;
20                 Obtain the state transitions of $x_n$ by inputting the time-series data into the Boolean function;
21                 Evaluate the fitness of $P_i$ using Eq. (9) ;
22             end
23             $Q_i' \leftarrow$ Replace $Q_i$ with the best one from $H(Q_i)$ ;
24         end
25         $P \leftarrow P'' \cup Q'$ ;
26         foreach chromosome $P_i$ in $P$ do
27             Decode the chromosome as a Boolean function;
28             Obtain the state transitions of $x_n$ by inputting the time-series data into the Boolean function;
29             Evaluate the fitness of $P_i$ using Eq. (9) ;
30         end
31     end
32     Output the optimal Boolean function of $x_n$;
33 end
34 Combine all the $N$ Boolean functions as the Boolean network;
```

$$F-score = \frac{2TP}{2TP + FP + FN} \tag{18}$$

where *TP* (true positive) and *FP* (false positive) represent the number of the correctly and incorrectly inferred interactions, respectively. *FN* (false negative) denotes the number of interactions that are absent in the inferred Boolean network but exist in the real Boolean network. *TN* (true negative) indicates the number of absent interactions in the inferred Boolean network which also does not exist in the real Boolean network. Among the five metrics, *F-score* is a comprehensive and crucial metric since that it trades off the two metrics: *Precision* and *Recall*.

As shown in Table 1, the information of the benchmark networks includes the number of nodes, the real in-degree K_{real}, the number of attractors N_{attr}, and the time-series data ($d \times \rho$). For each of the network, each time-series D_i ($i = 1, \cdots, d$) contains ρ time points. For the sake of validating the robustness against noise, the noise is introduced into the time-series data by randomly flipping the state of each node with the probability of ξ ($\xi \in \{0\%, 1\%, 5\%\}$). The five artificial Boolean networks are generated by BoolNet [31], with 50 nodes, 100 nodes, 120 nodes, 150 nodes, and 200 nodes, respectively. In addition, we utilize five real-world gene regulatory networks, including the idealized protein interaction network, the cAMP network [12], the Th-lymphocyte (or Th-cell) differentiation network [30], the Boolean network

Table 1
The parameters of the employed Boolean network models

Name	Nodes	K_{real}	Time-series ($d \times \rho$)	N_{attr}
Protein	6	2	10×10	3
cAMP	8	2	10×10	3
Th-cell	12	3	10×10	3
SSGLL	29	4	10×10	8
Drosophila	60	4	10×20	6
BN 50	50	4	10×20	30
BN 100	100	5	10×20	8
BN 120	100	5	10×20	10
BN 150	150	5	10×20	21
BN 200	200	5	10×30	24

Table 2
The parameter settings of GAPORE

Parameter	Meaning	Value
G	the maximum number of iterations	200
S	population size	300
E	the number of selected elites	10
pc	the probability of crossover	0.75
pm	the probability of mutation	0.7
λ	the l_2-norm regularization parameter	1.0
B	the number of Boolean rules	10

of survival signaling in large granular lymphocyte leukemia (SSGLL) [45], and the well-studied Drosophila segment polarity gene regulatory network [30]. To give an intuitive impression about the Boolean networks, two real-world gene regulatory networks are displayed below.

Fig. 4 shows the network topology of SSGLL and Eq.(19) is the Boolean functions of SSGLL. The Boolean network model of the Drosophila segment polarity gene regulatory network is widely considered as a single parasegment primordium of four identical cells, which has 15×4 = 60 nodes. In other words, the whole network is an interconnected network of four identical sub-networks. For the sake of saving space, Fig.5 demonstrates the intra-cellular regulatory interactions within a cell (i.e., cell 2) of the whole network and the inter-cellular regulatory interactions with cell 1 and cell 3. Nodes 16-30 inside the green dashed rectangle are in the cell 2. Nodes 3, 6, 7 are in the adjacent cell 1, and Nodes 33, 36, 37 are in the adjacent cell 3. Eq. (20) provides the Boolean functions for the cell 2.

For fair comparison, the parameter settings and tuning methods of the compared algorithms comply with the originally relative papers. All the four algorithms set the maximum in-degree as $K = 5$ due to the sparsity of networks [40]. In Table 2, the parameters of GAPORE are designed with the empirical values according to related standard genetic algorithms. Among these parameters, the l_2-norm regularization parameter λ plays a key role in overcoming the over-fit problem and improving the performance of GAPORE. Hence, it is necessary to comprehensively analyze the influence of λ on GAPORE, which is given in Section 4.1.

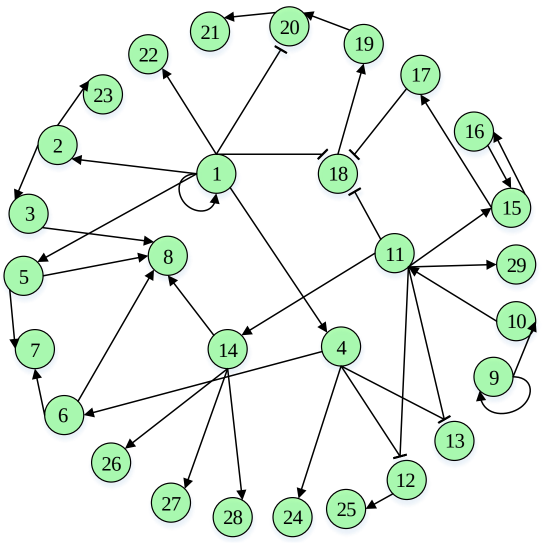


Figure 4: The network model of survival signaling in large granular lymphocyte leukemia with 29 nodes.

$$\begin{cases}
x_1(t+1) = x_1(t)\\
x_2(t+1) = x_1(t)\\
x_3(t+1) = x_2(t)\\
x_4(t+1) = x_1(t)\\
x_5(t+1) = x_1(t)\\
x_6(t+1) = x_4(t)\\
x_7(t+1) = x_5(t) \vee x_6(t)\\
x_8(t+1) = [x_6(t) \wedge (x_3(t) \vee x_5(t))] \wedge x_{14}(t)\\
x_9(t+1) = x_9(t)\\
x_{10}(t+1) = x_9(t)\\
x_{11}(t+1) = x_{10}(t)\\
x_{12}(t+1) = \neg[x_4(t) \vee x_{11}(t)]\\
x_{13}(t+1) = \neg[x_4(t) \vee x_{11}(t)]\\
x_{14}(t+1) = x_{11}(t)\\
x_{15}(t+1) = x_{11}(t) \vee x_{16}(t)\\
x_{16}(t+1) = x_{15}(t)\\
x_{17}(t+1) = x_{15}(t)\\
x_{18}(t+1) = \neg x_{17}(t) \vee [\neg x_1(t) \wedge \neg x_{11}(t)]\\
x_{19}(t+1) = x_{18}(t)\\
x_{20}(t+1) = \neg x_1(t) \wedge x_{19}(t)\\
x_{21}(t+1) = x_{20}(t)\\
x_{22}(t+1) = x_1(t)\\
x_{23}(t+1) = x_2(t)\\
x_{24}(t+1) = x_4(t)\\
x_{25}(t+1) = x_{12}(t)\\
x_{26}(t+1) = x_{14}(t)\\
x_{27}(t+1) = x_{14}(t)\\
x_{28}(t+1) = x_{14}(t)\\
x_{29}(t+1) = x_{11}(t)
\end{cases} \tag{19}$$

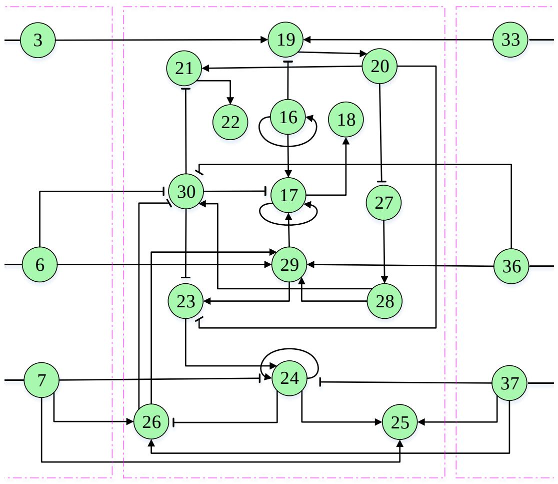


Figure 5: Part of the topology of the Drosophila segment polarity gene network. The edges (i.e., interactions) with → or ⊣ endpoints denote the activation or inhibition from the regulatory nodes to the target nodes, respectively.

$$\begin{cases} x_{16}(t+1) = x_{16}(t) \\ x_{17}(t+1) = [x_{29}(t) \wedge x_{16}(t) \wedge \neg x_{30}(t)] \\ \vee [x_{17}(t) \wedge (x_{29}(t) \vee x_{16}(t))] \wedge \neg x_{30}(t) \\ x_{18}(t+1) = x_{17}(t) \\ x_{19}(t+1) = \neg x_{16}(t) \wedge [x_3(t) \vee x_{33}(t)] \\ x_{20}(t+1) = x_{19}(t) \\ x_{21}(t+1) = \neg x_{30}(t) \wedge x_{20}(t) \\ x_{22}(t+1) = x_{21}(t) \\ x_{23}(t+1) = \neg x_{30}(t) \wedge \neg x_{20}(t) \wedge x_{29}(t) \\ x_{24}(t+1) = x_{23}(t) \vee [x_{24}(t) \wedge \neg x_7(t) \wedge \neg x_{37}(t)] \\ x_{25}(t+1) = [x_7(t) \vee x_{37}(t)] \wedge x_{24}(t) \\ x_{26}(t+1) = \neg x_{24}(t) \vee [x_7(t) \vee x_{37}(t)] \\ x_{27}(t+1) = \neg x_{20}(t) \\ x_{28}(t+1) = x_{27}(t) \\ x_{29}(t+1) = x_{28}(t) \wedge [x_{26}(t) \vee x_6(t) \vee x_{36}(t)] \\ x_{30}(t+1) = \neg x_{26}(t) \wedge \neg x_6(t) \wedge \neg x_{36}(t) \wedge x_{28}(t) \end{cases} \tag{20}$$

4.1. Approach validation and the regularization parameter selection

In order to validate the efficiency of the symbolic Boolean polynomial representation, the Drosophila segment polarity gene regulatory network is employed to compare the inference performance of three genetic algorithm-based network inference methods including MAGANI [25], GABP, and GAPORE. Among these genetic algorithms, MAGANI adopts the logical operation representation to express the unknown Boolean functions [25]. Both GABP and GAPORE utilize the symbolic Boolean polynomial representation to express the unknown Boolean functions. In addition, GABP only uses the genetic algorithm while GAPORE incorporates the local search strategy into the framework of the genetic algorithm. For fair comparison, the parameter settings and tuning methods of MAGANI comply with the originally relative paper. The maximum number of iterations for all algorithms is set as 200 and the population size is 300. Each algorithm runs ten times inferring the Drosophila segment polarity gene regulatory network under 0%, 1%, and 5% noise. The mean convergence curves obtained from ten runs are plotted in Fig. 6, where the black curves represent the results of MAGANI, the blue curves depict the results of GABP, and the pink curves illustrate the results of GAPORE. As shown in Fig. 6, MAGANI exists the premature convergence and tends to fall into a local optimum. This is because MAGANI with logical operation representation cannot represent or search the complex Boolean functions containing various logical operators with different precedence, such as the Boolean function of x_{17}. In contrast, GABP has better search capability since the symbolic polynomial representation can efficiently represent any complex Boolean rule by a coefficient combination. In addition, GAPORE shows evident superiority over GABP in terms of the convergence curves, which implies that the incorporation of the local search strategy can evidently promote the inference performance. Overall, the search capability can be improved using the polynomial representation of Boolean functions and incorporating the local search strategy into the genetic algorithm framework.

When it comes to selecting a proper regularization parameter λ, we apply GAPORE with different values of λ to inferring the artificial Boolean network with 100 nodes. Specifically, We change λ from 0 to 2 in the step of 0.2 in order to investigate the influence of λ on the performance of inferring the network topology. Fig. 7 shows the influence of the regularization parameter λ on the results with respect to *Recall*, *Precision*, and *F-score*. It is notable that GAPORE with $\lambda = 0$ achieves the highest values of *Recall* regardless of the noise ratio. With the increase of λ, the value of *Recall* gradually decreases. Meanwhile, the values of *Precision* increase sharply as λ increases from 0 to 0.8 and then reach a stable level when λ ranges from 1.2 to 2.0. This is due to the fact that GAPORE introduces the l_2-norm regularization to avoid the over-fit problem by restraining the in-degree K_n of a target node, which implies fewer regulatory nodes are recalled. Statistics on all the topology metrics are given in Table 3. Overall, the metric *Accuracy* remains stable when the λ increases from 0 to 2. Thus, we do not present the results with respect to *Accuracy* in the following experiments. Under 1% noise, *FPR* decreases from 0.016 (at $\lambda = 0$) to 0.002 (at $\lambda = 1$). The trend of *FPR* under 5% noise is similar to that of *FPR* under 1% noise. This phenomenon reflects that the regularization parameter λ can effectively reduce the number of false regulatory nodes. Obviously, the comprehensive performance analysis indicates that the value of λ should be limited within a reasonable region. The proper value of the regularization parameter λ ranges from 0.8 to 1.2.

In conclusion, it is effective and efficient for GAPORE to adopt the symbolic Boolean polynomial representation approach and design a proper l_2-norm regularization term, for the purpose of improving the inference performance.

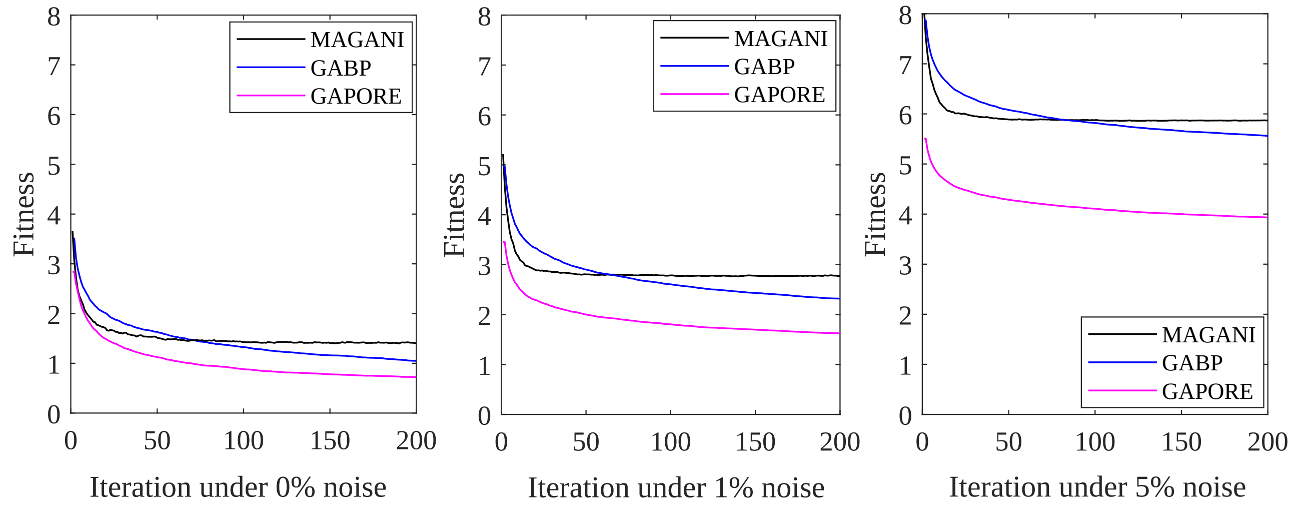


Figure 6: The convergence curves of MAGANI, GABP, and GAPORE on inferring the Drosophila segment polarity gene regulatory network under 0% noise, 1% noise, 5% noise, respectively.

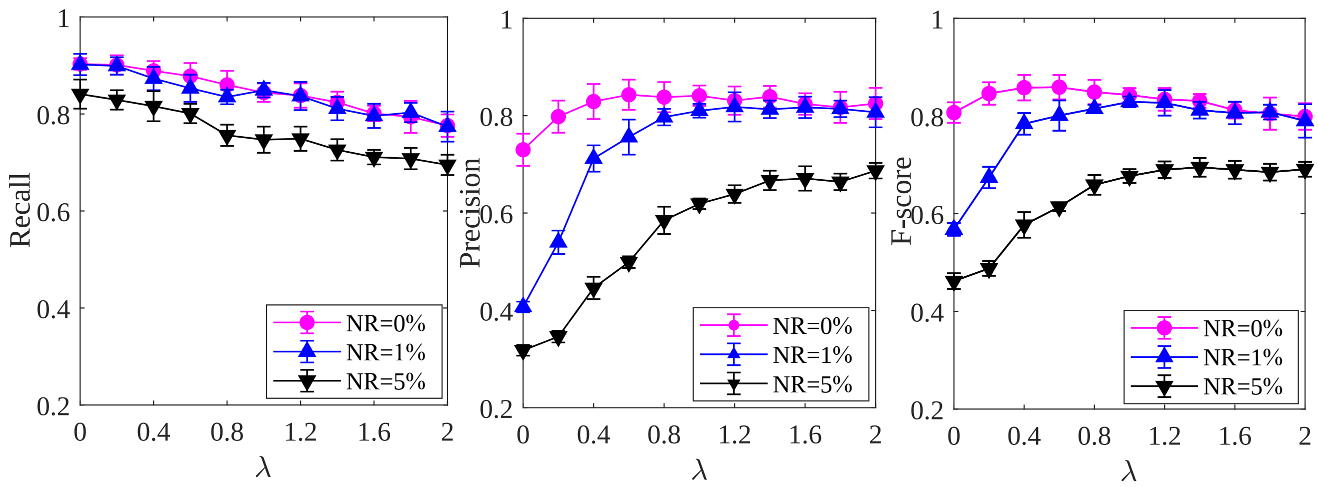


Figure 7: The influence of λ on the Recall, Precision, and F-score performance of GAPORE inferring the Boolean network with 100 nodes under the noise ratio NR= 0%, 1%, and 5%, respectively.

Table 3
The influence of λ on the performance of GAPORE inferring the Boolean network with 100 nodes under different noise ratios.

Noise	Evaluation	λ						
		0	0.4	0.8	1.0	1.2	1.6	2.0
0%	Accuracy	0.995 ± 0.001	0.996 ± 0.001	$\mathbf{0.996 \pm 0.001}$	0.996 ± 0.001	0.996 ± 0.001	0.995 ± 0.001	0.995 ± 0.001
	Recall	$\mathbf{0.903 \pm 0.012}$	0.889 ± 0.020	0.860 ± 0.029	0.844 ± 0.019	0.838 ± 0.025	0.801 ± 0.016	0.776 ± 0.023
	Precision	0.730 ± 0.033	0.829 ± 0.036	0.838 ± 0.031	$\mathbf{0.841 \pm 0.021}$	0.831 ± 0.029	0.824 ± 0.022	0.825 ± 0.032
	FPR	0.004 ± 0.001	0.002 ± 0.001	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	$\mathbf{0.002 \pm 0.000}$	0.002 ± 0.000
	F-score	0.807 ± 0.021	$\mathbf{0.858 \pm 0.026}$	0.849 ± 0.025	0.843 ± 0.014	0.834 ± 0.023	0.812 ± 0.017	0.799 ± 0.027
1%	Accuracy	0.983 ± 0.001	0.994 ± 0.001	0.995 ± 0.000	$\mathbf{0.996 \pm 0.000}$	0.996 ± 0.001	0.995 ± 0.001	0.995 ± 0.001
	Recall	$\mathbf{0.902 \pm 0.022}$	0.873 ± 0.024	0.835 ± 0.015	0.849 ± 0.015	0.837 ± 0.029	0.796 ± 0.025	0.774 ± 0.031
	Precision	0.407 ± 0.011	0.712 ± 0.027	0.797 ± 0.017	0.810 ± 0.014	$\mathbf{0.818 \pm 0.030}$	0.817 ± 0.022	0.807 ± 0.033
	FPR	0.016 ± 0.001	0.004 ± 0.001	0.003 ± 0.000	$\mathbf{0.002 \pm 0.000}$	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
	F-score	0.568 ± 0.013	0.784 ± 0.022	0.815 ± 0.008	$\mathbf{0.829 \pm 0.011}$	0.827 ± 0.026	0.806 ± 0.023	0.790 ± 0.034
5%	Accuracy	0.977 ± 0.001	0.986 ± 0.001	0.991 ± 0.001	0.992 ± 0.000	0.992 ± 0.000	0.993 ± 0.001	$\mathbf{0.993 \pm 0.001}$
	Recall	$\mathbf{0.841 \pm 0.030}$	0.816 ± 0.031	0.756 ± 0.022	0.747 ± 0.027	0.749 ± 0.025	0.711 ± 0.015	0.695 ± 0.021
	Precision	0.318 ± 0.011	0.446 ± 0.023	0.585 ± 0.028	0.619 ± 0.011	0.639 ± 0.018	0.671 ± 0.025	$\mathbf{0.687 \pm 0.016}$
	FPR	0.021 ± 0.001	0.013 ± 0.001	0.006 ± 0.001	0.006 ± 0.000	0.005 ± 0.000	$\mathbf{0.004 \pm 0.000}$	0.004 ± 0.000
	F-score	0.462 ± 0.016	0.577 ± 0.026	0.659 ± 0.020	0.677 ± 0.014	0.690 ± 0.017	0.690 ± 0.018	$\mathbf{0.691 \pm 0.015}$

4.2. Performance of inferring real-world gene regulatory networks

Table 4 demonstrates the performance of the four algorithms on inferring the real-world gene regulatory networks using the average and standard deviation values. Remarkably, GAPORE can infer the correct topology of the idealized protein interaction network under 1% noise. When the noise ratio increases to 5%, GAPORE can still infer the topology with *Recall*=1 and *F-score*=0.941. Interestingly, MIBNI can recall all the correct regulatory nodes and obtain the same results in terms of the four metrics under 1% and 5% noise. This is because MIBNI sets the fixed in-degree $K = 5$ for each node while the idealized protein interaction network is a very small network containing only 6 nodes. In terms of the cAMP network and the Th-cell network, GAPORE also outperforms the other three algorithms, which illustrates that GAPORE can achieve the best performance with a very high *Precision* when inferring the small real-world gene regulatory networks.

When it comes to inferring the SSGLL and Drosophila networks, the *F-score* results of MIBNI and Best-Fit decrease evidently as the network scale increases to $N = 29$ and $N = 60$, respectively. Since that the number of the possible Boolean functions of a node is $\left(\frac{2^{2^K} N!}{(N-K)!}\right)$, the search space for the candidate functions of a node increases to 6.1×10^{16} and 2.8×10^{18}, respectively. Thus, it is challenging to infer the whole Boolean network comprised of the Boolean functions of all nodes. In comparison to MIBNI and Best-Fit, ATEN still maintains better performance in terms of *F-score*. It is obvious that GAPORE outperforms ATEN almost on every metric except the *Recall* under 1% noise. Moreover, the *Precision* of GAPORE is noticeably higher than that of ATEN, which implies GAPORE is capable of handling the over-fit problem caused by the noise in time-series data.

Overall, GAPORE exhibits overwhelming superiorities over the other algorithms with respect to *Precision*, *FPR*, and *F-score* under both 1% and 5% noise. In terms of the metric *Recall*, GAPORE also outperforms the other algorithms in 7 out of 10 cases. As a result, it can be concluded that GAPORE has a significant advantage in inferring the topology of real-world networks over the state-of-the-art algorithms from the noisy time-series data.

4.3. Performance of inferring artificial Boolean networks

Table 5 shows the results of inferring artificial Boolean networks with different network scales, where the symbol '-' means that the results are not obtained within the wall time of five days. It is obvious that GAPORE can easily outperform the other three algorithms in terms of inferring the topology of large-scale Boolean networks from noisy time-series data.

In terms of the network BN 50 in Table 5, GAPORE is capable of inferring the most precise topology among the four algorithms under 1% noise. Although the noise ratio rises to 5%, GAPORE still keeps the best performance among these algorithms, which indicates GAPORE is robust against noise. MIBNI has the lowest *Precision* results because it can only infer the fixed number of regulatory nodes for every target node. The *Recall* results of Best-Fit are higher than that of ATEN while the *Precision* results of Best-Fit are much lower than that of ATEN. This is because Best-Fit uses the exhaustive search strategy and is more susceptible to the over-fit problem than ATEN.

When it comes to inferring the network BN 100, ATEN achieves *Precision*=0.652 under 1% noise which is better than that of MIBNI and Best-Fit. As can be observed from Table 3, the *Precision* of GAPORE under 1% noise increases from 0.407 (<0.652 of ATEN) to 0.810 (>0.652 of ATEN) as the regularization parameter λ increases from 0 to 1. From Table 5, ATEN achieves *Precision*=0.498 under 5% noise. Similarly, the *Precision* of GAPORE under 5% noise grows from 0.318 (<0.498 of ATEN) to 0.619 (>0.498 of ATEN) as the regularization parameter λ increases from 0 to 1. The phenomena illustrate that GAPORE can effectively improve the inference performance by adopting the l_2-norm regularization to address the over-fit problem.

Obviously, the size of search space increases sharply with the number of nodes in the Boolean network [40]. Specifically, the number of the candidate Boolean functions for a node increases from 3.8×10^{19} to 1.3×10^{21} with respect to the artificial Boolean network with $N = 100$ and 200, respectively. Thus, it is a huge challenge to infer large-scale Boolean networks from the noisy time-series data. As an exhaustive search algorithm, Best-Fit cannot infer the topology of the Boolean network with 200 nodes under 1% and 5% noise within the wall time of five days. From Table 5, MIBNI maintains the lowest *Precision* and *F-score* results on inferring these large-scale networks since that MIBNI sets the fixed in-degree for all nodes. Meanwhile, ATEN outperforms MIBNI in terms of the metrics *Precision* and *F-score* regardless of the network scale. This is because ATEN utilizes the And/Or tree ensemble method to enable the diversity of candidate solutions [39]. Although the search space increases dramatically as the network scale grows, GAPORE still outperforms the other algorithms on inferring the topology of the artificial network under both 1% and 5% noise. Remarkably, GAPORE achieves the highest *Precision* results and the lowest *FPR* results when inferring all the artificial Boolean networks, which implies that GAPORE is capable of inferring the Boolean functions precisely. One of the main reasons is that GAPORE introduces the dominant bit encoding scheme to enable the flexibility and diversity of solutions.

Notably, it is worth mentioning that GAPORE outperforms the other algorithms with respect to *F-score* when inferring every artificial network under both 1% and 5% noise, which indicates that GAPORE can maintain the balance of two metrics *Recall* and *Precision* by introducing the l_2-norm regularization to reduce the over-fit problem. In conclusion, GAPORE has the capability of inferring large-scale Boolean networks accurately.

Table 4
Performance on four real-world gene regulatory networks, respectively.

Networks	Algorithm	Noise 1%				Noise 5%			
		Recall	Precision	FPR	F-score	Recall	Precision	FPR	F-score
	ATEN	0.853 ± 0.010	0.851 ± 0.011	0.063 ± 0.005	0.852 ± 0.009	0.889 ± 0.013	0.782 ± 0.012	0.092 ± 0.006	0.832 ± 0.012
Protein	MIBNI	1	0.444	0.357	0.615	1	0.444	0.357	0.615
	Best-Fit	0.875	0.875	0.037	0.875	1	0.727	0.107	0.842
	GAPORE	1 ± 0.000	$\mathbf{1 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{1 \pm 0.000}$	1 ± 0.000	$\mathbf{0.889 \pm 0.000}$	0.036 ± 0.000	$\mathbf{0.941 \pm 0.000}$
	ATEN	0.909 ± 0.012	0.714 ± 0.003	0.075 ± 0.001	0.800 ± 0.004	0.727 ± 0.021	0.571 ± 0.016	0.113 ± 0.001	0.640 ± 0.011
cAMP	MIBNI	0.909	0.417	0.264	0.571	0.818	0.375	0.283	0.514
	Best-Fit	0.728	0.571	0.113	0.640	0.909	0.476	0.208	0.625
	GAPORE	$\mathbf{0.909 \pm 0.000}$	$\mathbf{1 \pm 0.000}$	$\mathbf{0 \pm 0.000}$	$\mathbf{0.952 \pm 0.000}$	$\mathbf{1 \pm 0.000}$	$\mathbf{0.839 \pm 0.021}$	$\mathbf{0.040 \pm 0.007}$	$\mathbf{0.912 \pm 0.013}$
	ATEN	0.955 ± 0.009	0.750 ± 0.008	0.057 ± 0.002	0.840 ± 0.010	0.864 ± 0.009	0.655 ± 0.012	0.082 ± 0.003	0.745 ± 0.011
Th-cell	MIBNI	0.636	0.389	0.180	0.483	0.591	0.361	0.189	0.448
	Best-Fit	**1**	0.786	0.049	0.880	**0.909**	0.606	0.106	0.727
	GAPORE	0.977 ± 0.024	$\mathbf{0.977 \pm 0.024}$	$\mathbf{0.004 \pm 0.004}$	$\mathbf{0.977 \pm 0.024}$	0.807 ± 0.032	$\mathbf{1 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.893 \pm 0.021}$
	ATEN	$\mathbf{0.821 \pm 0.011}$	0.516 ± 0.009	0.037 ± 0.005	0.634 ± 0.013	0.744 ± 0.009	0.439 ± 0.011	0.046 ± 0.003	0.553 ± 0.010
SSGLL	MIBNI	0.718	0.193	0.146	0.304	0.667	0.179	0.148	0.283
	Best-Fit	0.795	0.456	0.046	0.579	0.769	0.349	0.069	0.496
	GAPORE	0.746 ± 0.031	$\mathbf{0.857 \pm 0.032}$	$\mathbf{0.006 \pm 0.002}$	$\mathbf{0.798 \pm 0.029}$	$\mathbf{0.801 \pm 0.017}$	$\mathbf{0.787 \pm 0.014}$	$\mathbf{0.010 \pm 0.000}$	$\mathbf{0.794 \pm 0.011}$
	ATEN	0.628 ± 0.011	0.706 ± 0.009	0.012 ± 0.004	0.665 ± 0.009	0.490 ± 0.011	0.521 ± 0.013	0.023 ± 0.002	0.505 ± 0.007
Drosophila	MIBNI	0.438	0.210	0.068	0.284	0.320	0.153	0.074	0.207
	Best-Fit	**0.683**	0.498	0.045	0.576	0.564	0.327	0.059	0.414
	GAPORE	0.618 ± 0.035	$\mathbf{0.824 \pm 0.029}$	$\mathbf{0.006 \pm 0.001}$	$\mathbf{0.706 \pm 0.030}$	$\mathbf{0.514 \pm 0.032}$	$\mathbf{0.644 \pm 0.015}$	$\mathbf{0.012 \pm 0.002}$	$\mathbf{0.572 \pm 0.021}$

Table 5
Performance on artificial Boolean networks with 50 nodes, 100 nodes, 150 nodes, 200 nodes, respectively.

Networks	Algorithm	Noise 1%				Noise 5%			
		Recall	Precision	FPR	F-score	Recall	Precision	FPR	Fscore
	ATEN	0.884 ± 0.017	0.443 ± 0.015	0.029 ± 0.002	0.590 ± 0.016	0.925 ± 0.020	0.375 ± 0.017	0.038 ± 0.002	0.534 ± 0.014
BN 50	MIBNI	0.803	0.196	0.082	0.315	0.771	0.188	0.083	0.303
	Best-Fit	0.902	0.251	0.067	0.393	**0.934**	0.230	0.078	0.369
	GAPORE	$\mathbf{0.917 \pm 0.013}$	$\mathbf{0.879 \pm 0.015}$	$\mathbf{0.003 \pm 0.001}$	$\mathbf{0.897 \pm 0.010}$	0.802 ± 0.027	$\mathbf{0.719 \pm 0.023}$	$\mathbf{0.007 \pm 0.002}$	$\mathbf{0.758 \pm 0.019}$
	ATEN	$\mathbf{0.891 \pm 0.012}$	0.652 ± 0.016	0.006 ± 0.001	0.753 ± 0.019	$\mathbf{0.883 \pm 0.021}$	0.498 ± 0.016	0.012 ± 0.001	0.637 ± 0.011
BN 100	MIBNI	0.846	0.198	0.041	0.321	0.837	0.196	0.041	0.318
	Best-Fit	0.891	0.309	0.026	0.459	0.844	0.279	0.028	0.419
	GAPORE	0.849 ± 0.015	$\mathbf{0.810 \pm 0.014}$	$\mathbf{0.002 \pm 0.000}$	$\mathbf{0.829 \pm 0.011}$	0.747 ± 0.027	$\mathbf{0.619 \pm 0.011}$	$\mathbf{0.006 \pm 0.000}$	$\mathbf{0.677 \pm 0.014}$
	ATEN	$\mathbf{0.930 \pm 0.006}$	0.572 ± 0.008	0.007 ± 0.001	0.708 ± 0.014	$\mathbf{0.895 \pm 0.009}$	0.403 ± 0.014	0.013 ± 0.001	0.556 ± 0.012
BN 120	MIBNI	0.818	0.195	0.033	0.315	0.811	0.193	0.034	0.312
	Best-Fit	0.916	0.248	0.028	0.390	0.832	0.198	0.034	0.320
	GAPORE	0.838 ± 0.011	$\mathbf{0.780 \pm 0.024}$	$\mathbf{0.002 \pm 0.000}$	$\mathbf{0.807 \pm 0.018}$	0.716 ± 0.019	$\mathbf{0.633 \pm 0.020}$	$\mathbf{0.004 \pm 0.000}$	$\mathbf{0.672 \pm 0.014}$
	ATEN	$\mathbf{0.953 \pm 0.004}$	0.536 ± 0.008	0.006 ± 0.000	0.686 ± 0.012	$\mathbf{0.925 \pm 0.007}$	0.384 ± 0.009	0.012 ± 0.001	0.543 ± 0.011
BN 150	MIBNI	0.553	0.198	0.018	0.292	0.542	0.194	0.018	0.286
	Best-Fit	0.883	0.229	0.024	0.364	0.905	0.216	0.026	0.349
	GAPORE	0.809 ± 0.020	$\mathbf{0.799 \pm 0.014}$	$\mathbf{0.002 \pm 0.000}$	$\mathbf{0.804 \pm 0.015}$	0.720 ± 0.012	$\mathbf{0.597 \pm 0.020}$	$\mathbf{0.004 \pm 0.000}$	$\mathbf{0.653 \pm 0.016}$
	ATEN	0.664 ± 0.016	0.382 ± 0.012	0.006 ± 0.000	0.485 ± 0.013	0.617 ± 0.013	0.329 ± 0.023	0.008 ± 0.000	0.429 ± 0.017
BN 200	MIBNI	**0.824**	0.196	0.020	0.317	**0.802**	0.191	0.020	0.309
	Best-Fit	–	–	–	–	–	–	–	–
	GAPORE	0.782 ± 0.0268	$\mathbf{0.755 \pm 0.018}$	$\mathbf{0.001 \pm 0.000}$	$\mathbf{0.768 \pm 0.021}$	0.678 ± 0.023	$\mathbf{0.559 \pm 0.009}$	$\mathbf{0.003 \pm 0.000}$	$\mathbf{0.613 \pm 0.011}$

4.4. Performance on predicting the dynamics of networks

It is of significance to predict the dynamics of Boolean networks. As a discrete-time non-linear dynamical system, the Boolean network has several attractors that reflect its periodic behavior. In the context of biological systems, attractors characterize cellular phenotypes and give an insight into functional cellular states such as proliferation and apoptosis differentiation [32]. As a result, it is reasonable to evaluate the dynamics by comparing the number of the attractors retrieved by the algorithms [39].

Table 6 presents the number of attractors that can be predicted by the four inference algorithms. MIBNI is not capable of finding any attractors since MIBNI can infer only conjunctive or disjunctive Boolean functions [39]. The number of attractors N_{attr} for each benchmark network is list in Table 1. In terms of the real-world gene regulatory networks, ATEN retrieves more attractors than Best-Fit. Meanwhile, GAPORE outperforms the other algorithms in terms of retrieving the attractors under 1% noise. Especially, GAPORE obtains all the three attractors of the protein interaction network, the cAMP network, and the Th-cell network, respectively. When the noise ratio increases to 5%, GAPORE still retrieves the most attractors of the real-world gene regulatory networks. With regard to the artificial network BN 50, BN 100, BN 120, and BN 150, GAPORE achieves more attractors than both ATEN and Best-Fit under 1% and 5% noise, which implies that GAPORE can better predict the dynamics of these artificial networks than ATEN and Best-Fit. Notably, GAPORE retrieves all the attractors of BN 100

Table 6
The number of attractors retrieved by the four algorithms under 1% noise and 5% noise.

Noise	Algorithm	real-world networks					artificial networks				
		Protein	cAMP	Th-cell	SSGLL	Drosophila	BN 50	BN 100	BN 120	BN 150	BN 200
1%	ATEN	2	3	3	4	4	16	7	8	5	9
	MIBNI	0	0	0	0	0	0	0	0	0	0
	Best-Fit	2	2	3	2	2	7	4	4	0	0
	GAPORE	**3**	**3**	**3**	**6**	**5**	**24**	**8**	**10**	**13**	**19**
5%	ATEN	1	2	1	1	4	8	5	3	2	6
	MIBNI	0	0	0	0	0	0	0	0	0	0
	Best-Fit	1	1	1	1	1	5	2	2	0	0
	GAPORE	**2**	**3**	**2**	**6**	4	**13**	**7**	**9**	**8**	**14**

and BN 120 under 1% noise. When it comes to BN 150 and BN 200, only ATEN and GAPORE can still retrieve the attractors under 1% and 5% noise. In comparison to ATEN, GAPORE retrieves more attractors of BN 150 and BN 200 under 1% and 5% noise. Overall, it is obvious that GAPORE retrieves the most number of attractors with respect to both the real-world gene regulatory networks and the large-scale artificial Boolean networks. In conclusion, GAPORE outperforms the other algorithms in terms of predicting the dynamics of Boolean networks.

5. Conclusion

As a powerful and efficient model for gene regulatory networks, Boolean network inference has attracted many researchers' attention. However, it is still challenging to infer large-scale Boolean networks due to the noisy time-series data and lack of an efficient approach to representing unknown Boolean functions. To deal with large search space in many real-world Boolean network inference problems, we propose a novel inference algorithm called GAPORE to precisely infer the large-scale Boolean networks.

GAPORE is a hybrid evolutionary algorithm incorporating the local search into the genetic algorithm framework to enhance the search capability. Firstly, a symbolic polynomial representation method is proposed to efficiently represent the unknown Boolean function as the symbolic polynomial dynamical equation, which can describe any complex Boolean rules by only using the polynomial coefficient combinations. Secondly, the dominant bit encoding scheme is introduced to flexibly encode the symbolic polynomial dynamical equation, which both enables the flexibility of solutions and maintains the ease of implementation by varying the effective length instead of the total length of the chromosome. In addition, GAPORE successfully reduces the over-fit problem and shows overwhelming inference performance by introducing the l_2-norm regularization term and selecting a proper regularization parameter. Both the real-world gene regulatory networks and the artificial Boolean networks have been utilized to validate the performance of our proposed algorithm. Compared with the state-of-the-art algorithms, GAPORE is capable of inferring both the topology and the dynamics of the large-scale Boolean network accurately from the noisy time-series data.

Recently, probabilistic Boolean networks (PBNs) have been increasingly popular in that it enables more flexibility of model selection [29]. The PBN allows a random selection of Boolean functions for a target gene with a specific probability, which has shown promise in many areas. In the future, it is meaningful for us to extend GAPORE to the inference of PBNs.

Acknowledgements

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1701903 and in part by the National Natural Science Foundation of China under Grant 61973138. Shan He would like to thank the Southern University of Science and Technology (SUSTech) for the sabbatical visiting program.

References

[1] A, Y.Z., B, J.Z., C, W.X.A., D, J.C., 2019. Stabilization and oscillations design for a family of cyclic boolean networks via nodes connection. Neurocomputing 369, 61–68.

[2] Abbaspour, R.A., Samadzadegan, F., 2011. Time-dependent personal tour planning and scheduling in metropolises. Expert Systems with Applications 38, 12439–12452.

[3] Barman, S., Kwon, Y.K., 2017. A novel mutual information-based boolean network inference method from time-series gene expression data. Plos One 12, e0171097.

[4] Barman, S., Kwon, Y.K., 2018. A boolean network inference from time-series gene expression data using a genetic algorithm. Bioinformatics 34, i927–i933.

[5] Chen, K., Lv, Q., Lu, Y., Dou, Y., 2017. Robust regularized extreme learning machine for regression using iteratively reweighted least squares. Neurocomputing 230, 345–358.

[6] Cheng, D., Qi, H., Li, Z., 2011. Model construction of boolean network via observed data. IEEE Transactions on Neural Networks 22, 525–536.

[7] Cheng, D., Zhao, Y., 2011. Identification of boolean control networks. Automatica 47, 702–710.

[8] Faisal, S., Lichtenberg, G., Trump, S., Attinger, S., 2010. Structural properties of continuous representations of boolean functions for gene network modelling. Automatica 46, 2047–2052.

[9] Faisal, S., Lichtenberg, G., Werner, H., 2008. Polynomial models of gene dynamics. Neurocomputing 71, 2711–2719.

[10] Fu, Z.H., Hao, J.K., 2017. Knowledge-guided local search for the prize-collecting steiner tree problem in graphs. Knowledge-Based Systems 128, 78–92.

[11] Gambella, C., Ghaddar, B., Naoum-Sawaya, J., 2020. Optimization problems for machine learning: a survey. European Journal of Operational Research .

[12] Gao, B., Li, L., Peng, H., Kurths, J., Zhang, W., Yang, Y., 2013. Principle for performing attractor transits with single control in boolean networks. Physical Review E 88, 062706.

[13] Ghannami, A., Li, J., Hawbani, A., Al-Dubai, A., 2021. Stratified opposition-based initialization for variable-length chromosome shortest path problem evolutionary algorithms. Expert Systems with Applications 170, 114525.

[14] Gross, T., Wongchenko, M.J., Yan, Y., Blüthgen, N., 2019. Robust network inference using response logic. Bioinformatics 35, i634–i642.

[15] Huang, X., Yang, X., Zhao, J., Xiong, L., Ye, Y., 2018. A new weighting k-means type clustering framework with an l2-norm regularization. Knowledge-Based Systems 151, 165–179.

[16] Hutt, B., Warwick, K., 2007. Synapsing variable-length crossover: Meaningful crossover for variable-length genomes. IEEE transactions on evolutionary computation 11, 118–131.

[17] Jarrah, A.S., Laubenbacher, R., Stigler, B., Stillman, M., 2007. Reverse-engineering of polynomial dynamical systems. Advances in Applied Mathematics 39, 477–489.

[18] Kim, E., Ivanov, I., Dougherty, E.R., 2019. Quantifying the notions of canalizing and master genes in a gene regulatory network—a boolean network modeling perspective. Bioinformatics 35, 643–649.

[19] Krizhevsky, A., Sutskever, I., Hinton, G., 2012. Imagenet classification with deep convolutional neural networks. advances in neural information processing systems 25 (nips 2012) .

[20] Lähdesmäki, H., Shmulevich, I., Yli-Harja, O., 2003. On learning gene regulatory networks under the boolean network model. Machine learning 52, 147–167.

[21] Laubenbacher, R., Stigler, B., 2004. A computational algebra approach to the reverse engineering of gene regulatory networks. Journal of theoretical biology 229, 523–537.

[22] Lee, J., Kim, D.W., 2016. An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph. Information Sciences 332, 1–18.

[23] Li, H., Deb, K., Zhang, Q., 2019. Variable-length pareto optimization via decomposition-based evolutionary multiobjective algorithm. IEEE Transactions on Evolutionary Computation 23, 987–999.

[24] Li, Y., Chen, H., Zheng, J., Ngom, A., 2015. The max-min high-order dynamic bayesian network for learning gene regulatory networks with time-delayed regulations. IEEE/ACM transactions on computational biology and bioinformatics 13, 792–803.

[25] Liu, X., Shi, N., Wang, Y., Ji, Z., He, S., 2021. Data-driven boolean network inference using a genetic algorithm with marker-based encoding. IEEE/ACM Transactions on Computational Biology and Bioinformatics .

[26] Lynch, V.J., 2016. A copy-and-paste gene regulatory network. Science 351, 1029–1030.

[27] Martins, D.M.L., 2019. Reverse engineering database queries from examples: State-of-the-art, challenges, and research opportunities. Information Systems 83, 89–100.

[28] Maucher, M., Kracher, B., Kühl, M., Kestler, H.A., 2011. Inferring boolean network structure via correlation. Bioinformatics 27, 1529–1536.

[29] Melkman, A.A., Cheng, X., Ching, W.K., Akutsu, T., 2017. Identifying a probabilistic boolean threshold network from samples. IEEE transactions on neural networks and learning systems 29, 869–881.

[30] Menini, L., Possieri, C., Tornambè, A., 2019. Boolean network analysis through the joint use of linear algebra and algebraic geometry. Journal of theoretical biology 472, 46–53.

[31] Müssel, C., Hopfensitz, M., Kestler, H.A., 2010. Boolnet—an r package for generation, reconstruction and analysis of boolean networks. Bioinformatics 26, 1378–1380.

[32] Paulevé, L., Kolčák, J., Chatain, T., Haar, S., 2020. Reconciling qualitative, abstract, and scalable modeling of biological networks. Nature communications 11, 1–7.

[33] Pratapa, A., Jalihal, A.P., Law, J.N., Bharadwaj, A., Murali, T., 2020. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. Nature Methods 17, 147–154.

[34] Qiongbing, Z., Lixin, D., 2016. A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems. Expert Systems with Applications 60, 183–189.

[35] Ren, J., Hao, J.K., Rodriguez-Tello, E., Li, L., He, K., 2020. A new iterated local search algorithm for the cyclic bandwidth problem. Knowledge-Based Systems 203, 106136.

[36] Roy, G.G., Geard, N., Verspoor, K., He, S., 2020. Polobag: Polynomial lasso bagging for signed gene regulatory network inference from expression data. Bioinformatics .

[37] Shen, F., Liu, J., Wu, K., 2020a. Evolutionary multitasking fuzzy cognitive map learning. Knowledge-Based Systems 192, 105294.

[38] Shen, F., Liu, J., Wu, K., 2020b. A preference-based evolutionary biobjective approach for learning large-scale fuzzy cognitive maps: An application to gene regulatory network reconstruction. IEEE Transactions on Fuzzy Systems 28, 1035–1049.

[39] Shi, N., Zhu, Z., Tang, K., Parker, D., He, S., 2020. Aten: And/or tree ensemble for inferring accurate boolean network topology and dynamics. Bioinformatics 36, 578–585.

[40] Taou, N.S., Corne, D.W., Lones, M.A., 2018. Investigating the use of boolean networks for the control of gene regulatory networks. Journal of computational science 26, 147–156.

[41] Turky, A., Sabar, N.R., Dunstall, S., Song, A., 2020. Hyper-heuristic local search for combinatorial optimisation problems. Knowledge-Based Systems 205, 106264.

[42] Viswambaran, R.A., Chen, G., Xue, B., Nekooei, M., 2020. Evolving deep recurrent neural networks using a new variable-length genetic algorithm, in: 2020 IEEE Congress on Evolutionary Computation (CEC), IEEE. pp. 1–8.

[43] Wu, K., Liu, J., Chen, D., 2019. Network reconstruction based on time series via memetic algorithm. Knowledge-Based Systems 164, 404–425.

[44] Xie, W., Jia, X., Shen, L., Yang, M., 2019. Sparse deep feature learning for facial expression recognition. Pattern Recognition 96, 106966.

[45] Zhong, J., Li, B., Liu, Y., Lu, J., Gui, W., 2020. Steady-state design of large-dimensional boolean networks. IEEE transactions on neural networks and learning systems 32, 1149–1161.