scientific reports

OPEN

Dynamic Bayesian network structure learning based on an improved bacterial foraging optimization algorithm

Guanglei Meng[1,2], Zelin Cong[1,2]✉, Tingting Li[2], Chenguang Wang[2], Mingzhe Zhou[1,2] & Biao Wang[1,2]

With the rapid development of artificial intelligence and data science, Dynamic Bayesian Network (DBN), as an effective probabilistic graphical model, has been widely used in many engineering fields. And swarm intelligence algorithm is an optimization algorithm based on natural selection with the characteristics of distributed, self-organization and robustness. By applying the high-performance swarm intelligence algorithm to DBN structure learning, we can fully utilize the algorithm's global search capability to effectively process time-based data, improve the efficiency of network generation and the accuracy of network structure. This study proposes an improved bacterial foraging optimization algorithm (IBFO-A) to solve the problems of random step size, limited group communication, and the inability to maintain a balance between global and local searching. The IBFO-A algorithm framework comprises four layers. First, population initialization is achieved using a logistics-sine chaotic mapping strategy as the basis for global optimization. Second, the activity strategy of a colony foraging trend is constructed by combining the exploration phase of the Osprey optimization algorithm. Subsequently, the strategy of bacterial colony propagation is improved using a "genetic" approach and the Multi-point crossover operator. Finally, the elimination-dispersal activity strategy is employed to escape the local optimal solution. To solve the problem of complex DBN learning structures due to the introduction of time information, a DBN structure learning method called IBFO-D, which is based on the IBFO-A algorithm framework, is proposed. IBFO-D determines the edge direction of the structure by combining the dynamic K2 scoring function, the designed V-structure orientation rule, and the trend activity strategy. Then, according to the improved reproductive activity strategy, the concept of "survival of the fittest" is applied to the network candidate solution while maintaining species diversity. Finally, the global optimal network structure with the highest score is obtained based on the elimination-dispersal activity strategy. Multiple tests and comparison experiments were conducted on 10 sets of benchmark test functions, two non-temporal and temporal data types, and six data samples of two benchmark 2T-BN networks to evaluate and analyze the optimization performance and structure learning ability of the proposed algorithm under various data types. The experimental results demonstrated that IBFO-A exhibits good convergence, stability, and accuracy, whereas IBFO-D is an effective approach for learning DBN structures from data and has practical value for engineering applications.

Keywords Dynamic Bayesian networks, Structural learning, Swarm intelligence optimization algorithm, Bacterial foraging optimization algorithm

Bayesian networks (BNs) combine probability theory with graph theory and exhibit strong interpretability and high learning efficiency. The logical relationships inherent in data can be effectively investigated, offering a promising technical approach for establishing causal models for complex problems[1]. Dynamic Bayesian networks (DBNs) are powerful algorithmic tools that integrate the structure of static BNs with time-related information and are employed for dynamic uncertainty inference and temporal data analysis. DBNs have applications in various

[1]School of Automation, Shenyang Aerospace University, Shenyang 110136, China. [2]Aviation Science and Technology Key Laboratory of Air Combat System Technology, Shenyang 110136, China. ✉email: congzelin@stu.sau.edu.cn

fields, including artificial intelligence, machine learning, and automatic control[2]. Furthermore, DBNs have a broad range of engineering applications, such as managing transcriptional regulatory relationships between cancer genes[3], identifying connectivity issues between human brain regions through high-order DBNs using functional magnetic resonance imaging time series data[4], and analyzing the vascularization in the formation process of engineered tissues, aiming to enhance the accuracy of predicting future time steps and ensuring an acceptable uncertainty in forecasting the future progress of the organization[5]. Integrating structural prediction methods, such as mutual information and maximum information coefficient into the DBN model enhances the efficiency and scale of gene regulatory network reconstruction[6].

However, the introduction of time information into the DBN network increases search space complexity, reduces structure learning accuracy, and impedes the direct application of the static BN learning method. Early learning approaches involved constructing DBN network structures by experts based on their prior experience[7,8]. Conversely, for large datasets with numerous parameters and high complexity, relying only on expert experience poses challenges in establishing causal relationships based on temporal information and ensuring structural accuracy. To solve these problems, Leray et al.[9] proposed a classical DBN structure learning algorithm inspired by the dynamic max–min hill climb (DMMHC) local search algorithm. This algorithm allows for the quick identification of constraints based on temporal information in the 2T-BN model. Trabelsi et al.[10] introduced the heuristic greedy search (GS) algorithm[11], which was extended to 2T-BN networks to perform local optimal selection through a generative neighborhood algorithm as a way to expect to find the globally optimal DBN structure.

However, these classical DBN structure learning methods still have problems such as low efficiency of generating networks and lack of escaping local optimum mechanism. Therefore, how to further optimise the DBN structure learning methods to improve their search efficiency and search accuracy is the current research hotspot for scholars. In recent years, researchers have shown significant interest in nature-inspired metaheuristic algorithms based on swarm intelligence (SI) for optimization. Notable optimization algorithms include genetic algorithm (GA)[12], ant colony optimization[13], particle swarm optimization (PSO)[14,15], grey wolf optimization (GWO)[16], artificial bee colony[17], bat optimization[18], whale optimization[19], and firefly optimization[20]. Scholars have also improved the aspects of population initialization and individual optimization iterative updating strategies of these optimization algorithms on the original basis by introducing a series of effective methods such as Lévy flights[21], opposition-based learning[22], adaptive strategies, and hybrid algorithms, aiming to improve their optimization performance, as well as the accuracy and stability of the optimization results. Wu et al.[23] proposed an improved ant colony optimization algorithm (ICMPACO) by introducing multiple swarm strategies, co-evolutionary mechanism, pheromone updating strategy and pheromone diffusion mechanism, which improved the optimization ability and stability of the algorithm. And it was applied to Traveling Salesmen Problem to obtain better allocation results. Gao et al.[24] proposed an improved variable weight gray wolf optimization algorithm VW-GWO, which improves the probability of the algorithm escaping from the local optimum by introducing control parameters and decreasing control equations. Liu et al.[25] proposed an adaptive weighted particle swarm optimization algorithm (AWPSO) based on sigmoid function, which updated the acceleration coefficients using the sigmoid activation function of the neural network and considered both the particles to the global optimal position through an adaptive weighting strategy, thus improving the optimization efficiency and convergence characteristics of the algorithm. Zhang et al.[26] proposed a Chaotic Bacterial Foraging Optimization (ChaoticBFO) algorithm to achieve a reasonable balance between exploration and exploitation by introducing a chaotic initialization strategy and a chaotic local search with a "contraction" strategy in the convergence step. Mou et al.[27] proposed an adaptive non-dominated sorting genetic algorithm III (ANSGAIII), the algorithm enhances the objective function by considering non-linear relationships, equality constraints, actuator rate and position constraints. The algorithm solves the problem of autonomous berthing and dynamic positioning of over-actuated ships. Giri et al.[28] proposed an adaptive neighbourhood for locally and globally tuned biogeography based optimization algorithm (ANLGBBO) which inherits features of the nearest neighbour of the local best individual to be migrated along with a global best individual of the pool. And explore large Spaces by identifying areas with high-quality solutions. In addition, literature research shows that advanced optimization algorithms and their variants have played an important role in practical engineering applications in many intelligent fields, such as: Data mining engineering[29,30], automatic driving[31,32], intelligent robots[33], network topology architecture[34,35], military intelligent equipment[36,37] and other fields.

The use of metaheuristic search mechanisms to investigate the solution space is a common characteristic among SI-based algorithms. During each computation iteration, these algorithms update the state of the local models and generate new populations to search for local optima. The quality of the solutions continuously improves as the number of iterations increases, ultimately approaching the optimal model. Based on this, optimizing the DBN network structure using a high-performance swarm intelligence optimization algorithm is a novel and effective research method. Li et al.[38] introduced a binary PSO algorithm based on mutual information, which effectively prunes the search space. The algorithm accelerates the convergence speed of deep belief network structure learning by updating the particles using a probability threshold. Heng et al.[39] proposed a fitness function and redefined the encoding format of the particle swarm algorithm for DBN structure learning from incomplete datasets. Santos et al.[40] redefined the positions and velocities in the particle swarm algorithm to learn the high-order dynamic Bayesian network structure from large-scale and multivariate time series data. Jiang et al.[41] proposed an adaptive learning algorithm for DBN structure learning using a two-step strategy and adaptive crossover and mutation rates within a GA framework. Quesada et al.[42] proposed an order-invariant, Markov order-independent high-order particle encoding approach that exhibits scalability in high-order networks. Deng et al.[43] proposed an improved binary bat algorithm for learning the transitional network structure of a DBN and constructed a fitness function to quantitatively assess the node order in the network.

Although the aforementioned SI-based DBN structure learning approaches have achieved a certain level of effectiveness, there is still substantial potential for improvement in terms of optimization efficiency and accuracy.

Passino et al.[44] proposed a bacterial foraging optimization (BFO) algorithm that stimulates the foraging behavior of Escherichia coli bacteria in the human body. BFO is a global stochastic search algorithm, The simulation of the bacterial population comprises four steps: chemotaxis, grouping, reproduction, and elimination-dispersal. It has the characteristics of not requiring the gradient information of the optimization object during the optimization process, low complexity and fast convergence, which can be applied to reduce the number of iterative convergence times for finding the candidate network, jumping out of the local optimum, and searching for the highest scoring globally optimal DBN network structure. However, the original BFO also has certain defects, such as random steps of chemotactic activity, poor information exchangeability of clustering mechanism, and inability to maintain a balance between global and local search.

In summary, to enhance the BFO optimization performance, we propose a new hybrid algorithm, called the improved bacterial foraging optimization algorithm (IBFO-A), which aims to improve optimization iteration speed and accuracy while maintaining the low time complexity and fast convergence performance of BFO and balancing the global and local exploration and development capabilities. Then, within the framework of the IBFO-A algorithm, combined with a dynamic K2 scoring function and customized learning strategy, an IBFO-D method for DBN structure learning is designed to improve its ability to optimize the learning network structure from the data.

The main contributions of this study can be summarized as follows:

(1) To improve the population quality, the population is initialized using a logistics-sine chaotic mapping strategy. During the development and exploration phase of hybrid osprey optimization algorithm (OOA), the chemotactic activity of bacteria was reconstructed, improving the ability of individual bacteria to recognize and move toward the optimal target fitness value.

(2) Based on the replication idea of GA and the Multi-point crossover operator, the reproduction steps were reconstructed. This involves crossing the poor individual X_{worst} and fusing the better individual X_{best}, thereby improving performance, increasing the species diversity of the flora, and escaping the local optimal solution based on the elimination-dispersal operator.

(3) A dynamic K2 scoring function and V structure orientation rule are established. Combined with the IBFOA framework, the cumulative health score is saved during the breeding stage to reduce the number of iterative convergence in searching for candidate networks, and it is used in the elimination-dispersal stage to find the globally optimal DBN network structure with the highest score.

(4) The effectiveness of the proposed algorithms (IBFO-A and IBFO-D) is evaluated and comparisons are made with other mature algorithms through experimental tests on the benchmark data set. Statistical analyses of the experimental results are conducted as follows:

- Firstly, the IBFO-A algorithm is compared with seven other optimization algorithms using 10 sets of different types of CEC2005 benchmark functions (unimodal, multimodal, and hybrid). These include three original algorithms, two classical algorithms, and two recent advanced algorithms. Additionally, sensitivity analysis experiments were conducted for the three parameters of IBFO-A. The experimental results indicate that IBFO-A algorithm runs stably, ASR ranks first and has a certain competitiveness. Subsequently, we conducted comparative experiments on 12 optimization algorithms, including IBFO-A, using the CEC2019 benchmarking functions and two real-world engineering optimization problems. Some novel as well as improved optimization algorithms are included. The experimental results show that the IBFO-A algorithm exhibits good optimization performance, indicating its potential in real engineering applications.
- Secondly, the B_0 and $B_{\rightarrow}$ network structure learning capabilities of IBFO-D in non-temporal and temporal data samples are investigated, revealing that the generated network structure can converge stably within a high fitness value.
- Finally, IBFO-D is compared with two other structure learning algorithms using six 2T-BN temporal network data samples. The experimental results show that IBFO-D is an effective method for optimizing DBN structure learning from the data.

The remaining sections of this study are structured as follows. Preliminaries offers a review of the relevant concepts and scoring metrics of BN, along with an introduction to the basics of DBN. In Methodology, the principles of IBFO-A and IBFO-D algorithms and the design of dynamic scoring function are described in detail. Experimental section presents the results and analysis of the simulation experiments. Finally, Conclusion provides the conclusion and outlines plans for future research.

Preliminaries

Static Bayesian network

The Bayesian network N is represented as a binary tuple $N = (G, \Theta)$ comprising structure G and network parameters Θ. In graph theory, the independent relationships among a set of variables can be represented using a directed acyclic graph (DAG). Here, $G = (X, E)$ represents a specific instance or representation of such a graph; where X is a nonempty set of all nodes in the graph. $X = \{X_1, X_2, \ldots, X_i, \ldots, X_n\}$, X_i can be either an observed variable or a latent variable; E is the set of directed line segments between different variables in the DAG, and $X_j \rightarrow X_i$ d represents the direct dependencies between nodes[45].

$$E = \left\{X_j \rightarrow X_i | X_j \in pa(X_i), i = 1, \ldots, n\right\} \tag{1}$$

where $pa(X_i)$ is the "causes" of the node X_i, also called the set of parent nodes. Given the parent node set $pa(X_i)$, X_i is independent of its non-descendant node set $nd(X_i)$ based on Markov independence. Thus, the joint probability of several nodes X_i that follow the Markov rule can be expressed as follows:

$$P(X_1, \cdots, X_n) = \prod_{i=1}^{n} P(X_i | pa(X_i)) \tag{2}$$

The conditional probability table of each node X_i given its known parent node set $pa(X_i)$ is represented by the network parameter $\Theta = \{\Theta_1, \Theta_2, \ldots, \Theta_n\}$. It is possible to calculate the joint probability distribution of the node X_i when the network structure G and the network parameters Θ of a Bayesian network are known. Compared with other approaches for calculating joint probabilities, the efficiency of Bayesian network algorithms is significantly higher because of the conditional independence among nodes.

Scoring function

The search and score-based BN learning approach mainly comprises two parts: model selection and model optimization. Its core idea involves considering all possible structures as the domain, selecting a scoring function that assesses the quality of specific structures, and treating the process of identifying the best structure as an optimization problem of searching for the optimal value of the scoring function within the domain.

Prior knowledge about structure G is summarized as a probability distribution $P(G)$, referred to as the structure prior distribution for a Bayesian network $N = (G, \Theta)$. Similarly, prior knowledge about parameters Θ is summarized as another probability distribution $P(\Theta|G)$ referred to as the parameter prior distribution for a given structure G. In this manner, the prior distribution of N can be expressed as follows:

$$P(G, \Theta) = P(G)P(\Theta|G) \tag{3}$$

The posterior probability distribution $P(G|D)$ is calculated when given an observed dataset $D = \{D_1, D_2, \ldots, D_N\}$. Only the structural models G^* corresponding to the maximum posterior probability distribution in the search space are considered.

$$G^* = \arg\max_G P(G|D) \tag{4}$$

And

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)} \tag{5}$$

Selecting the structure with the maximum posterior probability is equivalent to selecting the structure that maximizes the following function since $P(D)$ does not depend on G:

$$\arg\min_G \log P(G|D) = \arg\min_G \log P(D|G) + \log P(G) + C \tag{6}$$

Based on penalized maximum likelihood or marginal likelihood, various scoring metrics, including Bayesian Dirichlet, Bayesian Dirichlet equivalent, K2, minimum description length, Bayesian information criterion, and mutual information test, have been proposed to assess the fitness of networks during the search process.

The most classic K2 scoring function formula is expressed as follows:

$$P(G, D) = P(G) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{7}$$

where n denotes the number of variables in the sample, q_i denotes the number of parent nodes for X_i, r_i denotes the number of possible values for X_i, N_{ijk} denotes the number of samples, and N_{ij} denotes the total number of samples.

Dynamic Bayesian networks

DBN is a graphical model structure that illustrates the conditional independence relationships between random variables and their temporal evolution patterns[46]. Its unique transition network can reflect the state changes of the system under different environmental factors in various time slices, showing the complex interactions and dependencies among variables in the system and offering a closer approximation to the real situations of dynamic multidimensional data.

However, representing $X_1, X_2, \ldots, X_n$ stochastic processes using DBN requires deriving a probability distribution over the random variable a, which can be highly complex. Thus, it is crucial to make appropriate assumptions about DBN and design a reasonable and efficient optimization algorithm for structure learning to study and model complex systems (see Methodology). These assumptions can be summarized as follows:

(1) The marginal directionality rule describes the dependency relationships between nodes in a finite time slice t, and the changes in conditional probabilities tend to converge to consistent stability across all processes.
(2) Given the random variables at time step t, the random variables at time step $t + 1$ are conditionally independent of the remaining random variables;$X_{t+1} \perp\!\!\!\perp (X_{t-1}, X_0) | X_t$. In other words, the Markov chain prop-

erty is satisfied to $P(X_{t+1}|X_0, X_1, \ldots, X_t) = P(X_{t+1}|X_t)$ by the entire dynamic discrete-time probabilistic process[47].

(3) Across all adjacent time steps, the network topology remains invariant and the transitional network, along with its corresponding conditional probability dependencies, remains the same. In other words, $P(X_{t+1}|X_t)$ is independent of time t.

The DBN constructed on the time trajectory of the random process comprises two components based on the aforementioned conditions: $(B_0, B_{\rightarrow})$.

(1) The initial network B_0, defined on the initial state X_0, and the joint probability distribution $P(X_0)$ obtained from it form the most initial graphical structure of the Bayesian network (BN) from which the prior probabilities of any node can be derived.
(2) The graphical structure of the BN composed of more than two time steps is represented by the transitional network $B_{\rightarrow}$, defined by variables X_0 and X_1, with transitional probabilities $P(X_{t+1}|X_t)$.

In other words, the entire DBN corresponds to $\{0, 1, 2, \ldots, T\}$ finite period, a and unfolds the probabilistic graphical model onto the topology of the random variable $X_0, X_1, \ldots, X_T$. The parent nodes of X_0 are those in the initial network B_0 at time 0. At time $t+1$, the parent nodes of X_{t+1} are those in the transitional network $B_{\rightarrow}$ that are relevant in both time steps t and $t+1$. A set of initial networks B_0, a transitional network $B_{\rightarrow}$, and a simple DBN model structure with two time slices are illustrated in Fig. 1.

To summarize, given a DBN model, the joint probability distribution on $X_0, X_1, \ldots, X_T$ is defined as follows:

$$P(X_0, X_1, \ldots, X_T) = P_{B_0}(X_0) \prod_{t=0}^{T} P_{B_{\rightarrow}}(X_{t+1}|X_t) \tag{8}$$

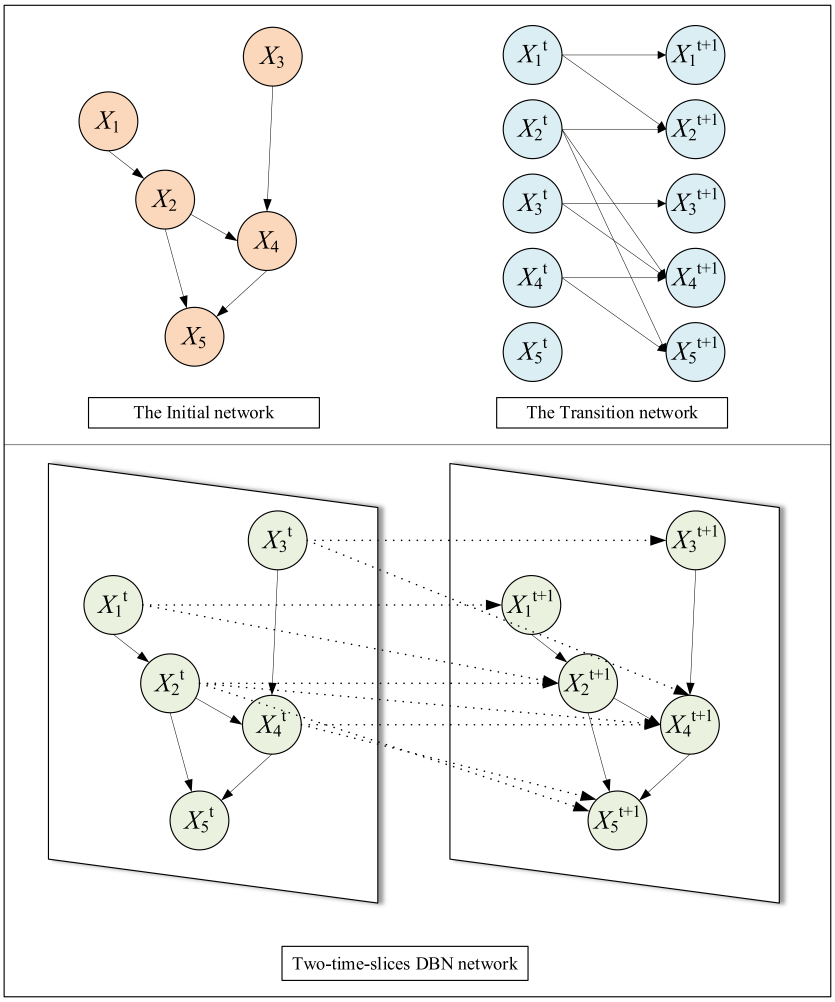

Figure 1. Dynamic Bayesian network model diagram.

To solve the actual optimization decision problem, the SI-based structure learning method of DBN extends the static optimization model, starting with the static initial network B_0. This process involves the construction of a basic graph model for dynamic intelligent optimization using time slice information. Figure 2 shows the specific algorithmic process. A round of the BN network node set can be generated through the transitional network $t + 1$ when the environmental factors change in round $B_{\rightarrow}$. A new population is generated along with the actual optimization problem based on the BN nodes in the round $t + 1$, which is then evolved and optimized to produce a set of excellent solutions and an optimal BN structure graph that matches the current environment, serving as the most suitable reasoning tool for the current problem. Subsequently, the node set to be optimized in the round $t + 2$ is generated by the DBN, and this process continues. As environmental factors change, inference and optimization are conducted to effectively address various emergencies and enhance the mitigation of the effect of uncertain factors on the findings.

Methodology

IBFO-A optimization algorithm

Population initialization

For swarm intelligent optimization algorithms, the selection of individual initial positions often affects the algorithm's iterative convergence performance. The original BFO algorithm uses random initial locations, which results in the dispersion of most bacteria generated at the initial moment being far away and even not meeting the boundary constraints. Chaotic mapping[48] is an effective method to improve the population initialization of the optimization algorithm. In this study, a logistics-sine mixed method proposed by Demir et al.[49] is used to integrate the two most universal methods of chaotic mapping: Logistic mapping and Sine mapping evenly distribute the population in the mapping space, significantly improving the species diversity and search efficiency of the population in the following ways:

The upper bound of the feasible domain of each dimension of the objective function $ub = [ub_1, ub_2, \ldots, ub_d]$ and the lower bound $lb = [lb_1, lb_2, \ldots, lb_d]$. The location matrix modeling of bacterial individuals in the search space is as follows:

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,j} & \cdots & x_{n,m} \end{bmatrix}_{n\times m} \tag{9}$$

Use the logistics-sine method to initialize the bacterial individual location:

$$a_{i+1} = ra_i(1 - a_i) \tag{10}$$

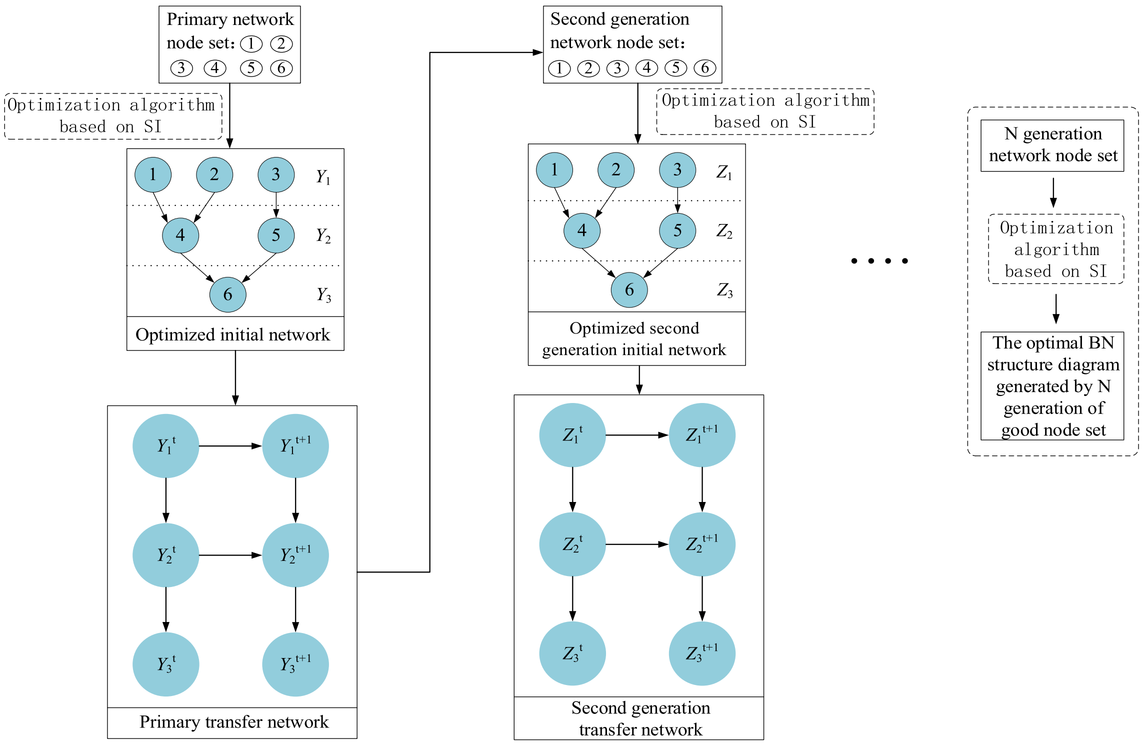
Primary network node set: ① ② ③ ④ ⑤ ⑥
Optimization algorithm based on SI
Y_1
Y_2
Y_3
Optimized initial network
Y_1^t Y_1^{t+1} Y_2^t Y_2^{t+1} Y_3^t Y_3^{t+1}
Primary transfer network
Second generation network node set: ① ② ③ ④ ⑤ ⑥
Optimization algorithm based on SI
Z_1
Z_2
Z_3
Optimized second generation initial network
Z_1^t Z_1^{t+1} Z_2^t Z_2^{t+1} Z_3^t Z_3^{t+1}
Second generation transfer network
N generation network node set
Optimization algorithm based on SI
The optimal BN structure diagram generated by N generation of good node set

Figure 2. DBN intelligent optimization model diagram.

$$b_{i+1} = \frac{(4 - r)\sin(\pi b_i)}{4} \tag{11}$$

$$x_{i+1} = (a_{i+1} + b_{i+1})(\bmod 1) \tag{12}$$

where $a_i \in (0, 1)$ and $b_i \in (0, 1)$ are randomly generated series, a_{i+1} represents the logistic chaotic mapping, b_{i+1} represents the sine chaotic mapping, r represents the chaos coefficient, and x_{i+1} is the bacterial chaotic mapping value determined by logistics-sine.

Finally, the chaotic sequence is mapped to the solution space:

$$X_i = lb + x_{i+1}(ub - lb) \tag{13}$$

Chemotactic activity

Chemotactic activity plays a crucial role in the IBFO-A's algorithm, in which bacteria first tentatively choose the direction for a "flip" motion, and then swim to a nutrient-rich area through a "swim" motion. In the original BFO algorithm, the trend activity is randomly given the i th bacterial movement step $C(i)$ and receives the attraction signal from other individuals in the population to swim to the center of the population, and the attraction between bacteria is represented by $\mathrm{J^i_{cc}}\left(\theta, \theta^{\mathrm{i}}(\mathrm{j,k,1})\right), \mathrm{i} = 1, 2, \ldots, \mathrm{S}$. At the same time, there will be repulsion between bacteria, which prevents the consumption of nearby nutrients by maintaining a certain distance. $\mathrm{J_{cc}}(\theta, \mathrm{P(j,k,l)})$ said that the combined influence of attraction and repulsion between bacteria is considered at the same time, and its computation formula is as follows:

$$\begin{aligned}
\mathrm{J_{cc}}(\theta, \mathrm{P(j,k,l)}) &= \sum_{\mathrm{i=1}}^{\mathrm{s}} J^i_{CC}(\theta, \theta^i(\mathrm{j,k,l})) \\
&= \sum_{\mathrm{i=1}}^{\mathrm{s}} \left[-\mathrm{d_{atact}}\ \exp\left(-\mathrm{w_{attact}} \sum_{\mathrm{m=1}}^{\mathrm{p}} \left(\theta_{\mathrm{m}} - \theta^{\mathrm{i}}_{\mathrm{m}}\right)^2\right)\right] \\
&\quad + \sum_{\mathrm{i=1}}^{\mathrm{s}} \left[\mathrm{h_{repenatint}}\ \exp\left(-\mathrm{w_{repelinat}} \sum_{\mathrm{m=1}}^{\mathrm{p}} \left(\theta_{\mathrm{m}} - \theta^{\mathrm{i}}_{\mathrm{m}}\right)^2\right)\right]
\end{aligned} \tag{14}$$

where $\mathrm{P(j,k,l)}$ represents the position of each bacterium in the population S after the j trend operation, the k replication operation, and the l elimination-dispersal operation, $\theta = [\theta_1, \ldots, \theta_D]^T$ is a point on the optimization domain, θ^i_m is the m element of the i bacteria, dattract represents the amount of attraction released by the bacteria, Wattract is used to measure the width of the attraction signal. hrepenatint indicates the amount of rejection released by the bacteria, and Wrepelinat measures the width of the rejection signal.

However, there are some problems in the original BFO trend activity. First, the bacterial movement step $C(i)$ is given randomly, resulting in a low convergence accuracy of the algorithm. To solve this problem, most scholars choose to design a new step size. Supriyono et al.[50] developed three types of step size strategies: linear step size, quadratic step size, and exponential step size. Niu et al.[51] proposed a linear chemotactic decline step and a nonlinear chemotactic decline step as well as other types of non-adaptive steps[52–54].

In addition, the effect of communication between bacterial groups is limited, but the clustering mechanism with complex objective function cannot effectively guide bacterial individuals to the high-nutrient (fitness value) region, resulting in the algorithm often falling into the local optimal value prematurely. To solve this problem, scholars often choose to ignore the original clustering mechanism and combine better communication mechanisms to improve the algorithm. Chen et al.[55] combined the PSO algorithm to enhance intercellular communication and proposed an adaptive foraging strategy using area-focused search. Wang et al.[56] also chose to combine the PSO algorithm and Gaussian distribution to adjust the chemotactic activity of the flora and strengthen the ability of information exchange among the populations. Zhao et al.[57] employed the gravitational mechanism in GSA to improve the ability of information exchange between individuals in the chemotactic step of the BFO algorithm.

The Osprey optimization algorithm was proposed by Mohammad Dehghani and Pavel Trojovsky in 2023 to simulate the predation behavior of Osprey[58]. In the first stage of OOA, the Osprey identifies the position of the fish (fitness value) and performs the arrest (moving in the direction of high fitness and updating the individual position). For each Osprey, the position of the other Osprey with a better target fitness value in the search space is also regarded as the fish school. OOA Phase 2 brings the fish to the appropriate position to feed (moving in a random direction and updating the individual position). Among them, FP_i operator with certain clustering and optimal value searching ability in the OOA algorithm, and $x^{P1}_{i,j}$ operator with more optimal positions to update individual positions, can be used to solve the problems in the BFO algorithm. In summary, this study chose to combine the first phase of OOA with BFO chemotaxis to fully improve the performance of the IBFO-A algorithm. The second stage and subsequent elimination-dispersal activities are not selected to be combined with this stage.

The mathematical formula of "flip" movement:

$$FP_i = \{X_k | k \in \{1, 2, \ldots, N\} \wedge F_k < F_i\} \cup \{X_{best}\} \tag{15}$$

Formula 15 is used to investigate the search space with a good target value, where FP is a set of i target locations and XB is the best candidate solution.

The mathematical formula of "swimming":

$$x_{i,j}^{P1} = x_{i,j} + r_{i,j} \cdot \left(SF_{i,j} - I_{i,j} \cdot x_{i,j}\right) \tag{16}$$

$$X_i = \begin{cases} X_i^{P1}, \ F_i^{P1} < F_i \\ X_i, \quad else \end{cases} \tag{17}$$

Formula 16 calculates the new position of the individual, and if this new position improves the target fitness value, the previous position is replaced according to Formula 17, where X_i is the original position of the individual, $x_{i,j}^{P1}$ is the new position of the individual i, F_i^{P1} is its objective function value, $SF_{i,j}$ is the candidate solution chosen by the individual i, $r_{i,j}$ is the random number in the interval [0,1], and $I_{i,j}$ is the random number in the set $\{1, 2\}$.

Reproductive activity
Reproduction is a fundamental biological behavior observed in various species and is a crucial aspect of life preservation. With each reproductive generation, the search efficiency of the colony improves. It has been mentioned in the conventional BFO algorithm that inferior bacteria should be eliminated and superior bacteria retained for reproduction. The cumulative state of the health fitness value of the i bacteria is represented by $J^i = \sum_{j=1}^{Nc+1} J\left(i, j, k, l\right)$.

The population of bacteria is divided into two sets based on the accumulation of their health fitness scores: X_{best}, comprising the top-ranked elite individuals with higher cumulative scores and X_{worst}, comprising the lower-ranked inferior individuals with lower cumulative scores. Then, a "genetic" approach was employed to perform reproduction activities following the specific formula below:

$$X_{\text{best}} = X'_{\text{best}} \tag{18}$$

$$X_{\text{co}} = X'_{\text{best}} \otimes X_{\text{worst}} \tag{19}$$

$$X' = X_{\text{co}} \oplus X_{\text{best}} \tag{20}$$

where $\otimes$ represents a crossover operator employed to perform the "Multi-point crossover"[59] of the encoding.$\oplus$ denotes the fusion operator. X'_{best} represents the reproduction elite individual obtained by X_{best} after replication and coding. X'_{best} and X_{worst} perform crossover operations on their encodings, leading to a single reproductive crossover individual X_{co}. X' denotes the new bacterial population obtained by fusing with X_{co} and X_{best}. This method enables an increase in the chemotactic ability of bacteria with lower cumulative scores, i.e., it enhances the average quality of the entire population while maintaining the original total number of bacterial individuals S. Furthermore, it improves the species diversity of the population and prevents the algorithm from becoming trapped in local optima.

Elimination-dispersal activity
In this study, the elimination-dispersal activity mechanism was enhanced based on adaptation theory. The bacterial population randomly selects and performs elimination-dispersal activity after each N_r round of reproduction, prompting the bacterial individuals to produce new solutions and conduct a new search for positions, thereby escaping local optima.

The specific definition of the elimination-dispersal function is expressed as follows:

$$P_{ed}^{i} = \frac{\left|f_{\min}\right| + \left|(f_{\max} - f_i) * P_{ed}^{i-1}\right|}{\left|f_{\min}\right| + \left|(f_{\max} - f_{\min}) * P_{ed}^{i-1}\right|} \tag{21}$$

where P_{ed}^{i} is the elimination-dispersal probability at the current moment P_{ed}^{i-1} represents the probability of elimination-dispersal at the previous time.$f_{\min}$ represents the worst goal score in history, $f_{\max}$ represents the best goal score in history, and f_i represents the current goal score of specific bacteria i.

With an increase in the number of iterations, the adaptive elimination-dispersal probability shows a nonlinear decreasing trend. At the early stage of iteration, to explore the solution space more widely, a larger elimination-dispersal probability is needed to find other foraging paths. In the later iteration, due to the guidance of the global optimal solution, the algorithm conducts a fine search near the global optimal solution, and the elimination-dispersal probability is reduced. Thus, the local development ability is enhanced, and the bacteria can find the target solution more quickly and accurately. In addition, for formula (21), to determine a better objective function, the population elimination-dispersal probability increases when the current score is close to the lowest score.

K2 dynamic scoring function

K2 scoring function differs from that of static BN because of the introduction of time information in DBN. A dynamic scoring function is necessary to measure the validity of the network structure. Thus, the K2 dynamic scoring function in IBFO-D is discussed in this section.

First, the initial network B_0 can be learned from the dataset assuming that the training set consists of N complete sequence samples, where the length of the l-th sample is N_l, and a specific value is assigned to the variable $x^l[0], \ldots, x^l[N_l]$. Then, the transitional network $B_\rightarrow$ can be learned from the transformed data $N = \sum N_l$. Considering the definitions of network parameters $t = 1, \cdots, T$ and sequence samples θ within the time slice N:

$$\begin{aligned} \theta^0_{i,j,k} &= P\left(X_i[0] = k_i | Pa\left(X_i[0] = j_i\right)\right) \\ \theta^{\rightarrow}_{i,j,k} &= P\left(X_k[t] = k_i | Pa(X_k[t]) = j_i\right) \end{aligned} \tag{22}$$

$$\begin{cases} N^0_{i,j,k} = \sum\limits_l I\left(X_i[0] = k_i, Pa\left(X_i[0] = j_i; x^l\right)\right. \\ N^{\rightarrow}_{i,j,k} = \sum\limits_l \sum\limits_0 I\left(X_i[t] = k_i, Pa(X_i[t]) = j_i; x^l\right) \end{cases} \tag{23}$$

where I represents an indicator function, specifically defined as:

$$I = \begin{cases} 1 \text{ if } \left(X_i[t] = k_i, Pa(X_i[t]) = j_i\right) \\ 0 \; else \end{cases} \tag{24}$$

The joint probability density of DBN is expressed as follows:

$$P_{\text{DBN}}(x[0], x[1], \ldots, x[T]) = P_{B_0}(X[0]) \prod_{t=0}^{T-1} P_{B_\rightarrow}(x[t+1] | x[t]) \tag{25}$$

The structure of DBN decomposes the likelihood function distribution into:

$$P(D|G) = \int P(D|G, \theta) P(\theta|G) \mathrm{d}\theta \tag{26}$$

The first term of the integral is decomposed into the following formula:

$$P(D|G, \theta) = \prod_i \prod_j \prod_k \left(\theta^0_{i,j,k}\right)^{N^0_{i,j,k}} \cdot \prod_i \prod_j \prod_k \left(\theta_{i,j,k}\right)^{N^{\rightarrow}_{i,j,k}} \tag{27}$$

The second term of the integral is decomposed into the following formula, assuming that the prior distribution on conditional probabilities is conditionally independent:

$$P(\theta|G) = \prod_i \prod_j P\left(\theta^0_{i,j,k}\right) \cdot \prod_i \prod_j P\left(\theta^{\rightarrow}_{i,j,k}\right) \tag{28}$$

The likelihood function can be rewritten as the product of two integrals by substituting the aforementioned formula:

$$P(D|G) = \prod_i \prod_j \int \prod_k \left(\theta^0_{i,j,k}\right)^{N^0_{i,j,k}} \times P\left(\theta^0_{i,j,k}\right) \times \mathrm{d}\theta^0_{i,j,k} \tag{29}$$

The likelihood function $P(D|G)$ can be further expressed as a product of $K2_0$ and $K2_\rightarrow$ given the hyperparameter $N^0_{i,j,k} N^{\rightarrow}_{i,j,k}$ and complete data, and with the parameter prior following a Dirichlet distribution:

$$P(D|G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N^0_{i,j} + r_i - 1)!} \cdot \prod_{k=1}^{r_i} N^{\rightarrow}_{i,j,k}! \tag{30}$$

where n denotes the number of variable samples, q_i denotes the number of parent nodes for X_i, r_i denotes the number of possible values for X_i, $N^0_{i,j}$ denotes the number of samples, and $N^{\rightarrow}_{i,j,k}$ denotes the total number of samples.

This study selects the logarithm of the likelihood function to minimize in practical applications. The final expression of the dynamic K2 scoring formula is as follows:

$$\begin{aligned} Score(G|D) &= K2_0(G|D) + K2_\rightarrow(G|D) \\ &= \sum_{i=1}^{n} q_i \cdot \log(r_i - 1)! - \sum_{j=1}^{q_i} \log\left(N^0_{i,j} + r_i - 1\right)! + \sum_{k=1}^{r_i} \log\left(N^{\rightarrow}_{i,j,k}\right)! \end{aligned} \tag{31}$$

IBFO-D optimization algorithm

Initialization

In Swarm intelligent optimization algorithms, encoding approaches to generate abstract structures and the concretization of the optimization process are crucial elements. In this study, the network structure is represented using an adjacency matrix $A = (a_{ij})$ with $n \times n$ dimensions. The directed edge from node i to node j is represented by $a_{ij} = 1$, whereas the absence of a connection between node i and node j is denoted by $a_{ij} = 0$.

$$a_{ij} = \begin{cases} 1 & i \text{ is a parent of } j \\ 0 & \text{no edges or deleted edges} \end{cases} \tag{32}$$

The specific process first initializes an empty adjacency matrix, develops and explores the search space through bacteria in the direction of high fitness, and constantly updates the location if and only if the new location has a higher score. Then, the network graph structure is updated by directional rules, and the process is repeated until the network structure with the highest K2 score in the time slice t is found, and the optimization iteration of the $t+1$ time slice is started. The DBN structure is represented as a DAG. Thus, it is crucial to consider their validity when constructing the initial network. In other words, in the searched DBN, the network structure should not contain any cycles or bidirectional edges. Reflected in the adjacency matrix, this indicates that the nodes in the matrix should not form cycles and that elements symmetrically located about the diagonal should not be to 1. Figures 3 and 4 illustrate examples of generating initial network B_0 and transfer network $B_{\rightarrow}$ structural adjacency matrices, respectively. Node labels can be simplified to further reduce the search space, leading to $B_0 = 11|01|00$ and $B_{\rightarrow} = 00110|00011|00001|00010|00001|00000$.

Network structure learning

IBFO-D In the framework of the IBFO-A algorithm, the state relationship between nodes in the DBN was considered. Restrictions were imposed on the tendency directions of the bacteria, such as N_S and N_C, considering the state relationships between nodes in the DBN. Consequently, three edge orientation rules, namely "add edge," "remove edge," and "reverse edge," were designed:

(1) Add Edge: Given a collection of nonempty nodes $X = \{x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_j | x_i \in G, x_i \notin \prod(x_j)\}$, if $e_{ij} = x_i \rightarrow x_j$ is added and $e_{ij} \in G$ holds, then $G' \in G \cup (e_{ij})$;

(2) Remove Edge: Given a directed edge set $E = \{e_{ij} = x_i \rightarrow x_j | x_i \in \prod(x_j), i = 1, \cdots, n\}$, if e_{ij} is removed, then $G' \in G \backslash (e_{ij})\, X = \{x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_j | x_i \in G, x_i \notin \prod(x_j)\}$" and $e_{ij} = x_i \rightarrow x_j$ are removed;

(3) Reverse Edge: Given a directed edge set $E = \{e_{ij} = x_i \rightarrow x_j | x_i \in \prod(x_j), i = 1, \ldots, n\}$, if e_{ij} is removed, $e_{ji} = x_j \rightarrow x_i$ is added, and $e_{ji} \in G$ holds, then $G' \in G \backslash (e_{ij}) \cup (e_{ji})$.

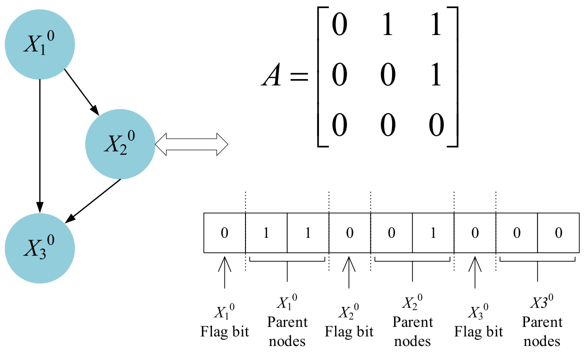

Figure 3. Initial network B_0.

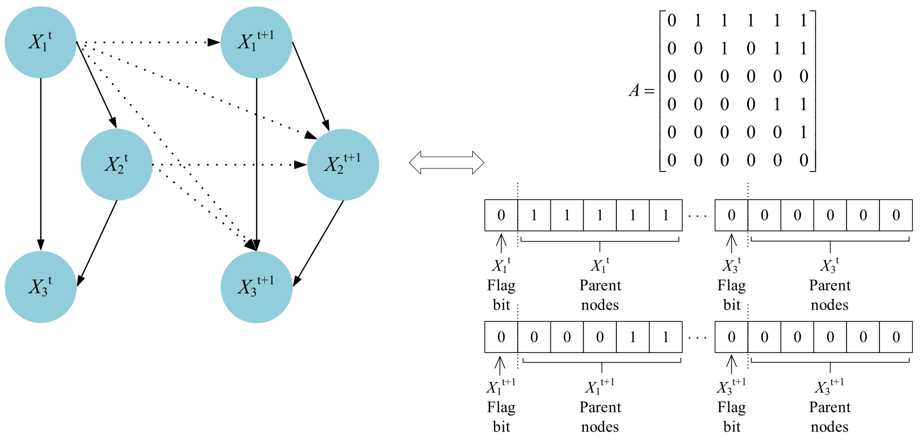

Figure 4. Transfer network $B_{\rightarrow}$.

The chemotactic activity process continued until the bacteria reached a fixed position and no longer moved or had moved the maximum number of chemotaxis, which corresponded to finding the network structure with the highest K2 score in the DBN network or reaching the upper limit of search iterations.

To choose healthy bacteria, it is essential to assess the health level of each bacterium. This assessment involves computing the sum of the fitness values of the chemotaxis steps. A higher cumulative value signifies that maximum nutrition has been obtained, making it more suitable for reproduction. In this study, the step fitness value of bacteria is defined based on the dynamic scoring function in DBN structure learning. Specifically, the health score for the i bacterium is expressed as follows:

$$HS_t^i = K2_{0(G|D)}(i,j,k,l) + \sum_{j_t=1}^{N_c} K2_{\rightarrow(G|D)}(i_t,j_t,k_t,l_t) \tag{33}$$

$K2_{0(G|D)}(i,j,k,l)$ is defined as the prior fitness value function for the i bacterium during the j chemotaxis, k reproduction, and l elimination-dispersal when generating the initial network. $K2_{\rightarrow(G|D)}(i_t,j_t,k_t,l_t)$ is the fitness value function for subsequent transition networks. Health function assesses the accumulated K2 score for individual bacteria throughout the entire process of chemotaxis operations.

To determine the global optimal network structure, the specific definition of the elimination-dispersal function is expressed as follows:

$$P_{ed}^t = \frac{|HS_{\min}| + |(HS_{\max} - HS_t^i) * P_{ed}^{t-1}|}{|HS_{\min}| + |(HS_{\max} - HS_{\min}) * P_{ed}^{t-1}|} \tag{34}$$

where P_{ed}^t is the elimination-dispersal probability at the current moment. P_{ed}^{t-1} is the elimination-dispersal probability of the previous moment. $HS_{\min}$ is the lowest historical health score, $HS_{\max}$ is the highest historical health score, and HS_t^i is the current specific health score of bacterium i.

Algorithm description

The IBFO-D Algorithm proposed in this study is shown in Algorithm 1. The whole DBN structure learning process is summarized as follows: the algorithm starts from the initialization of network parameters, randomly generates the initial DAG population, and finds the high-quality network structure through the chemotactic activity formula (15–17). At the same time, the driving force of DBN local optimization is generated according to three operators. The fitness value of each bacterium, namely the K2 score, was calculated, and the cumulative value HS_t^i was recorded as a health score. By selecting elite individuals with high HS_t^i, the average optimization ability of the bacterial population was updated according to the formula (18–20) to improve the convergence speed, while preserving certain species diversity to prevent falling into local optimality. According to formula (34), determine whether the bacterial individual generates a new solution and re-searches. According to the above optimization steps, as well as the dynamic K2 scoring measures and constraints, until a high-score network structure matching the data set is searched.

```
Input: Dataset $D$
Output: Dynamic Bayesian network and score
Initialization:
  a). Set parameters:
    $N$ : Bacterial population size、 $T$ : Maximum time step of the DBN、 $Ne$ : Number of elimination-dispersal
    $Nr$ : Number of reproduction、 $Nc$ : Number of chemotaxis、 $Ns$ : Number of swimming
    $pc$ : Crossover probability、 $p_{ed}$ : Elimination-dispersal probability
  b). Initialize the bacterial population
  for i = 1 to $N$
    Create the position of each bacterial is represented by $x_i$ . The fitness value of $f(x_i)$ is calculated according to formula. (31). The best
    position of each bacterial is represented by $x_{best}$ . Structure $G$ is determined according to $x_{best}$ , $f(x_i)$ and orientation rules.
  end
1   for time step = 1 to $T$
2       for elimination-dispersal loop: 1 to $Ne$
3           for reproduction loop: 1 to $Nr$
4               for chemotaxis loop: 1 to $Nc$
5                   Calculate the fitness score $J = f(x_i)$ for the current $G(i)$
6                   Find the best fitness score $J_{\min}$ and best position $X_{best}$
7                   for swim loop: 1 to $Ns$
8                       The current fitness $J$ and population location $x_i$ set were updated by formula (16) and (17)
9                       if $J_{\min} < J$ , then update the directed graph $G(i)$ according to the orientation rule
10                          let $J_{\min} = J$ , $G_{best} = G(i)$
11                  end
12              end
13              if ( $p \le pc$ ) then go to Step 3.
14              else The current health value of bacterial community was calculated according to formula (33)
15                  After all bacterial individual codes are arranged in ascending order, cross compounding is
16                  performed according to formulas (18-20)
17          end
18          The elimination-dispersal probability is calculated according to formula (34)
19          if ( $p \le p_{ed}$ ) then go to Step2
20          else Perform elimination-dispersal operations to randomly update the location of the bacterial population
21      end
22  end
23  Return $G_{best}$ and $J_{\min}$
```

Algorithm 1. IBFO-D

Experimental section

Experimental preparation

In the test experiment, we mainly focus on the optimization performance of the proposed method and its learning effect on the DBN model structure, and do not deny the validity and novelty of other optimization algorithms and their modeling in the corresponding domain. The parameter values of each algorithm used in the experiment are shown in Table 1.

To test the optimization performance of the proposed algorithm, Firstly, we use IBFO-A and seven other optimization algorithms to perform comparative test experiments on 10 different benchmark functions. The reference function comes from CEC2005[60], including the multi-peak function, single-peak function, and fixed-dimensional multi-peak function, which is used to test the convergence speed, accuracy, effectiveness, and global search ability of the algorithm. The specific reference function is shown in Table 2. Then we select three kinds of hyperparameters for parameter sensitivity analysis to test the stability of the algorithm.

Secondly, We conducted several more sets of comparative experiments of optimization algorithms. These include the eight optimization algorithms mentioned above and some novel optimization algorithms as well as improved optimization algorithms. The benchmark function for the simulation experiments is from CEC2019, which is a popular benchmark function that is quite effective in testing the performance of optimization algorithms. Then we use 2 sets of real-world engineering optimization problems to test the optimization performance of the algorithms from multiple perspectives. The CEC2019 specific reference function is shown in Table 3.

Finally, for the IBFO-D algorithm network learning performance test, various dynamic benchmark network experiments were selected. These dynamic benchmark networks were derived from two well-known static BN networks: the Asia and Alarm networks. These 2T-BN networks comprise two time slices and include an initial network represented as B_0 and a transition network represented as $B_{\rightarrow}$. Benchmark network data can be obtained from the provided Supplementary URL online.

Method	Population size	Maximum generation	Other parameters
IBFO-A	60	1000	$r = 0.86; P_{ed} = 0.8; pc = 0.2$
BFO	60	1000	$P_{ed} = 0.8; \Delta \in [-1, 1]$
OOA	60	1000	$r \in [0, 1]; I = \{1, 2\}$
GA	60	1000	$pc = 0.8; pm = 0.05$
GWO	60	1000	$a \in [2, 0]; \|r_1\| \in [0, 1]; \|r_2\| \in [0, 1]$
PSO	60	1000	$w_{\max} = 0.9; w_{\min} = 0.6; v_{\max} = 6; c_1 = 2; c_2 = 2$
BWO	60	1000	$r_1 \in (0, 1); r_2 \in (0, 1); \beta = 1.5$
DBO	60	1000	$k \in (0, 0.2]; b \in (0, 1)$
COA	60	1000	$temp = rand * 15 + 20$
GO	60	1000	$\text{pro} \in (0, 1); \text{rnd} \in (0, 1); \text{coe} \in (0, 1)$
AWPSO	60	1000	$w_{\max} = 0.9; w_{\min} = 0.4; v_{\max} = 0.5; v_{\min} = -0.5$
ChaoticBFO	60	1000	$\mu = 0.8; P_{ed} = 0.8; \Delta \in [-1, 1]$

Table 1. Parameter settings for the algorithms used.

Test function	Category	D	S	$f_{\min}$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	Unimodal test functions	30	$[-100, 100]^n$	0
$f_2(x) = \sum_{i=1}^{n} \|x_i\| + \prod_{i=1}^{n} \|x_i\|$		30	$[-10, 10]^n$	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$		30	$[-100, 100]^n$	0
$f_4(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	Multimodal test functions	30	$[-5.12, 5.12]^n$	0
$f_5(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$		30	$[-600, 600]^n$	0
$f_6(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_{i+1})\right] + (x_n - 1)\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$		30	$[-50, 50]^n$	0
$f_7(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	Multimodal test functions with fix dimension	4	$[-5, 5]^n$	3.075e−4
$f_8(x) = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$		2	$[-2, 2]^n$	3
$f_9(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]$		6	$[0, 1]^n$	− 3.32
$f_{10}(x) = -\sum_{i=1}^{10} \left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$		4	$[0, 10]^n$	− 10

Table 2. CEC2005 benchmark functions used in the experimental study. where D is the dimension of the function, $f_{\min}$ is the minimum value of the function, and search range $S \subseteq R^n$.

The experimental setup comprised the following environment: Windows 11 operating system, MATLAB and Python programming language, 32.0 GB RAM, an Intel Core i7-12700 K CPU running at 5.0 GHz, and an NVIDIA GeForce RTX 3080Ti graphics card.

Experimental results and analysis of IBFO-A compared with other SI algorithms

Comparison and analysis of eight optimization algorithms in CEC2005 benchmark functions

The experimental design of algorithm optimization performance comparison is as follows: The optimization algorithm in IBFO-A is compared with three original optimization algorithms, BFO, OOA, and GA; two classical optimization algorithms, GWO and PSO, two recent advanced optimization algorithms BWO[61] and DBO[62], and a total of eight optimization algorithms are compared and tested on 10 groups of different types of benchmark functions. We uniformly set experimental parameters for all optimization algorithms, in which the population size is set to $N = 60$, the maximum number of iterations $T = 1000$, and the upper bound ub, lower bound lb, dimension D, and optimal value $f_{\min}$ of different test functions are set as shown in Table 2. We present 10 sets of optimization convergence curves, specific scores, and running timelines, and record the average score ranking (ASR) and average run-time ranking (ATR) of the algorithm that runs 30 times independently (if both algorithms converge to the optimal value, the ASR is determined by the number of iterations).

The simulation results in Table 4 show that IBFO-A can converge to the optimal value for 6 of the 10 benchmark functions. In unimodal and multimodal functions, IBFO-A converges directly to the optimal value 0 on the F1, F2, F3, F4, and F5 functions. In addition, it can converge directly to the optimal value 3 on the F8 function, and it is also very close to the theoretical optimal value in other fixed-dimensional multi-peak test functions, which shows that it has good global optimization ability.

Here, we choose the original BFO, OOA, and GA algorithms as reference objects. According to the analysis in Fig. 5, the GA and BFO algorithms perform poorly on F1, F2, and F3 unimodal functions, BFO improves somewhat on multi-modal functions F4 and F5, and the OOA algorithm performs better. IBFO-A can stably converge to the optimal with less than half of the OOA iterations. GA still performs poorly in F7 and F9 functions of fixed dimension, and the convergence values of OOA and BFO are also different from the theoretical optimal values. Compared with the IBFO-A algorithm, the performance of the IBFO-A algorithm is competitive. From the experimental data in Table 4 and the convergence curve in Fig. 5, except for poor performance on the F6 generalized penalized function, compared with the other seven algorithms, IBFO-A exhibits the best performance in the seven function scoring tests, and ASR ranks first. This shows that the IBFO-A algorithm has good convergence speed and accuracy, and proves that the improved chemotactic step and the replication step using cross strategy can avoid falling into the local optimal solution and enhance the local search ability of the algorithm. In addition, the ATR of the IBFO-A algorithm ranks fourth. From the perspective of algorithm time complexity, the time complexity of IBFO-A and BFO is $O(n)$, whereas that of OOA is $O(n^2)$. Therefore, the computation time of IBFO-A is better than that of OOA. However, due to the extra computing steps, it consumes more computing time than the classical BFO, PSO, and GWO, which is also a limitation of the algorithm in this study.

Sensitivity analysis

This section discusses the hyperparameter sensitivity analysis of IBFO-A algorithm. We selected three hyperparameters that mainly affect the optimization performance of IBFO-A algorithm, including population size N, elimination-dispersal probability P_{ed}, and crossover probability pc. We set four different parameter values for each hyperparameter to be discussed, and use the IBFO-A algorithm to optimize several typical test functions of CEC2005 under these parameter settings. The specific parameter values and results are shown in Figs. 6, 7, 8 and Table 5.

From the sensitivity analysis of IBFO-A to hyperparameter N, it can be seen that with the increase of population size, the probability of finding the global optimal solution will increase, thus improving the optimization performance of the algorithm. However, large population sizes can also lead to increased time costs. P_{ed} determines the probability of initial elimination-dispersal occurrence of individual bacteria. According to the sensitivity analysis of IBFOA to hyperparameter P_{ed}, high P_{ed} parameter value enables bacteria to explore the

Test function	Description	D	S	$f_{\min}$
F1	Storn's Chebyshev polynomial fitting problem	9	$[-8192, 8192]$	1
F2	Inverse Hilbert matrix problem	16	$[-16384, 16384]$	1
F3	Lennard–Jones minimum energy cluster	18	$[-4, 4]$	1
F4	Rastrigin's function	10	$[-100, 100]$	1
F5	Grienwank's function	10	$[-100, 100]$	1
F6	Weierstrass funetion	10	$[-100, 100]$	1
F7	Modified Schwefel's function	10	$[-100, 100]$	1
F8	Expanded Schaffer's F6 function	10	$[-100, 100]$	1
F9	Happy Cat function	10	$[-100, 100]$	1
F10	Ackley function	10	$[-100, 100]$	1

Table 3. CEC2019 benchmark functions used in the experimental study.

	IBFO-A		BFO		OOA		GA		GWO		PSO		BWO		DBO	
	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time
F1	0	0.34518	3.8826e−4	0.304	0	0.37524	7.2368e−03	0.35948	1.1303e−152	0.27802	1.1153e−10	0.11689	0	0.49239	0	0.39853
F2	0	0.35709	1.3612	0.26638	0	0.37985	0.025851	0.38608	3.2631e−84	0.27342	3.9462e−06	0.13724	1.0557e−245	0.60831	5.7691e−179	0.4057
F3	0	0.50364	0.19572	0.26274	0	0.74867	3692.011	0.605	7.4765e−78	0.40169	7.6957e−05	0.2619	0	0.65241	0	0.80969
F4	0	0.306	0	0.24253	0	0.31499	3.0364	0.35543	0	0.29823	1.9909	0.13859	0	0.46038	0	0.38498
F5	0	0.37711	0	0.25357	0	0.45344	1.0136	0.40474	0	0.32898	0.15506	0.18492	0	0.52517	0	0.43327
F6	7.1435e−11	0.88702	1.3959e−05	0.25567	4.7116e−32	1.486	9.3882e−05	0.90122	4.7202e−08	0.73673	6.992e−13	0.61882	4.9019e−07	1.0446	4.7358e−32	0.90217
F7	3.0753e−04	0.37903	2.0981e−3	0.19671	1.1452e−3	0.39969	2.2021e−2	0.39899	2.0363e−2	0.23916	9.8938e−4	0.09926	3.0749e−04	0.45177	1.2232e−3	0.4013
F8	3	0.30494	3	0.24963	3	0.31474	3	0.33	3	0.16243	3	0.060188	3.2	0.36315	3	0.30783
F9	− 3.2374	0.42087	− 3.0711	0.39666	− 2.993	0.41104	− 1.9379	0.38357	− 3.203	0.25917	− 3.2031	0.12689	− 3.3136	0.45415	− 3.322	0.36787
F10	− 10.5364	0.58074	− 4.0465	0.29268	− 10.5363	0.69076	− 2.2012	0.5037	− 10.5363	0.35811	− 10.5364	0.26637	− 10.5364	0.59392	− 5.1756	0.50704
ASR	1		6		2		8		4		7		5		3	
ATR	4		2		7		5		3		1		8		6	

Table 4. Comparison results of IBFO-A, BFO, OOA, GA, GWO, PSO, BWO, and DBO on the 10 benchmark functions.

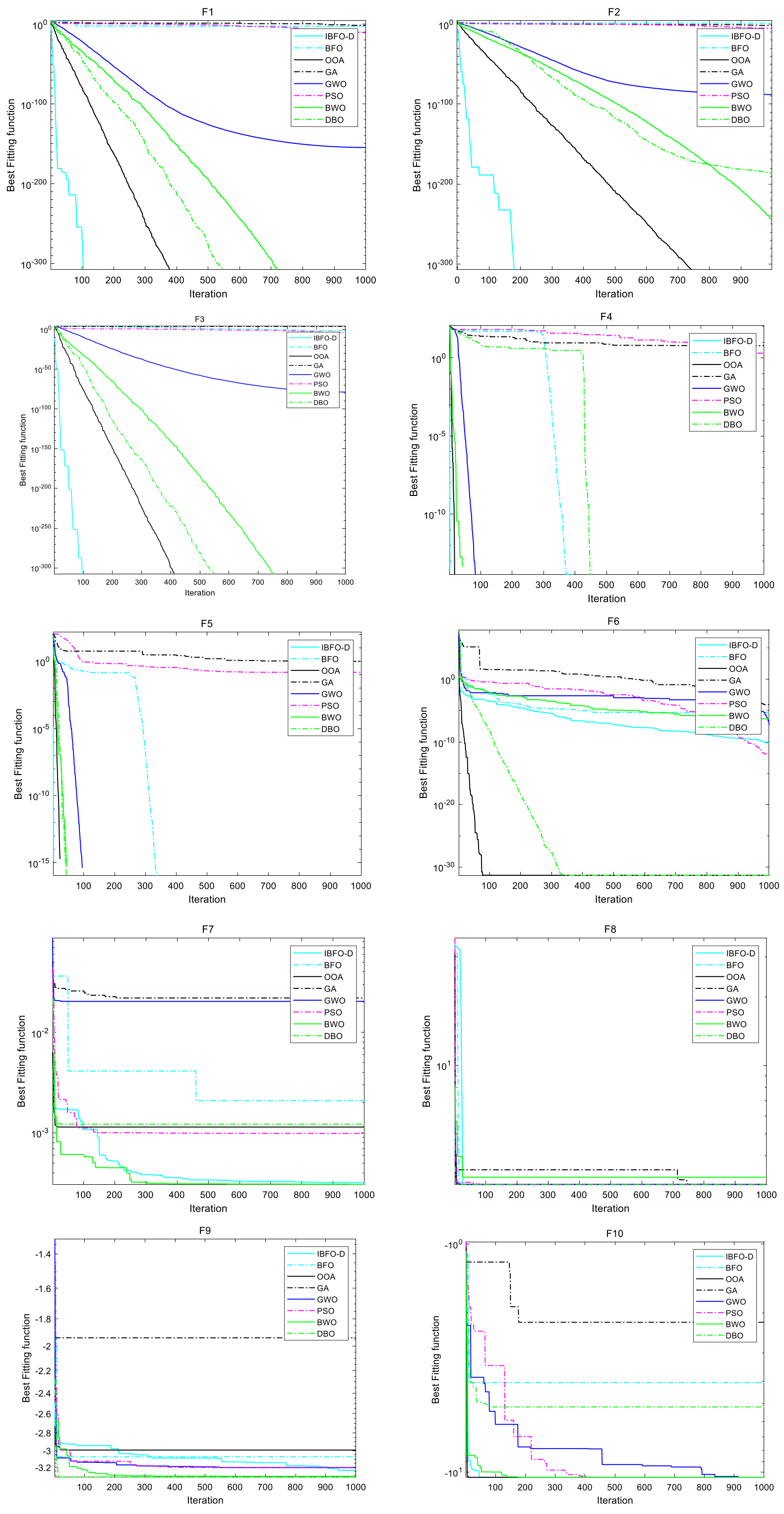

Figure 5. Comparison of convergence curves of 8 optimization algorithms.

solution space more extensively in the early stage of iteration, which increases the possibility of quickly searching for the global optimal solution. In addition, the improved adaptive activity solves the problem that excessive elimination-dispersal probability in the late iteration will lead to frequent update of bacterial colony location, which makes it difficult to conduct fine search near the optimal solution. In addition, it can be found that IBFO-A algorithm can also search the global optimal solution when P_{ed} parameter value is low, but it needs more iterations and time cost. From the sensitivity analysis of IBFO-A to hyperparameter pc, it can be seen that the higher the probability of pc, the more the coding composition of the individual in the flora is affected by other individuals,

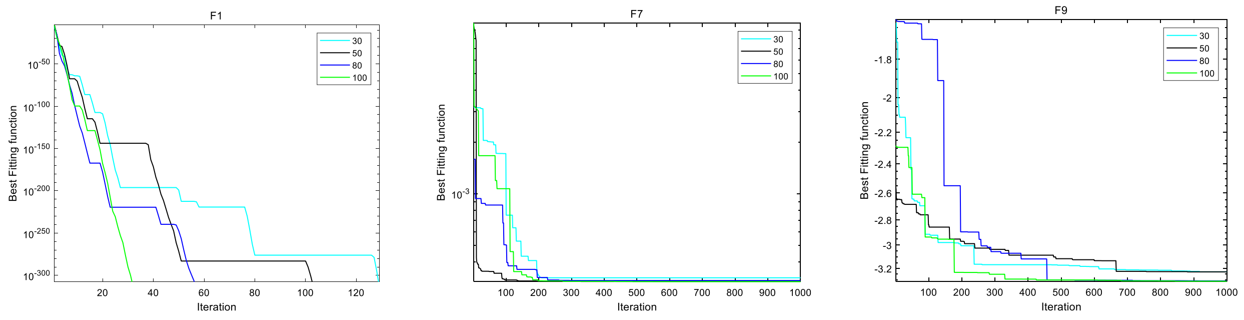

Figure 6. Sensitivity analysis of IBFO-A to parameter *N*.

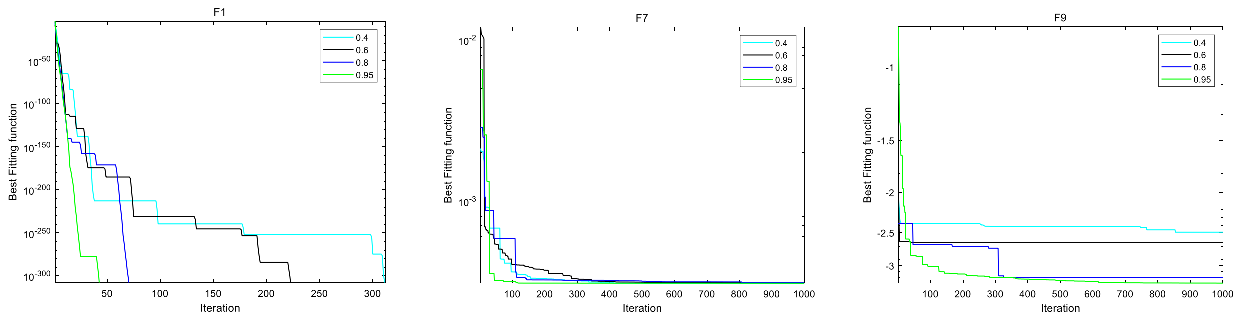

Figure 7. Sensitivity analysis of IBFO-A to parameter P_{ed}.

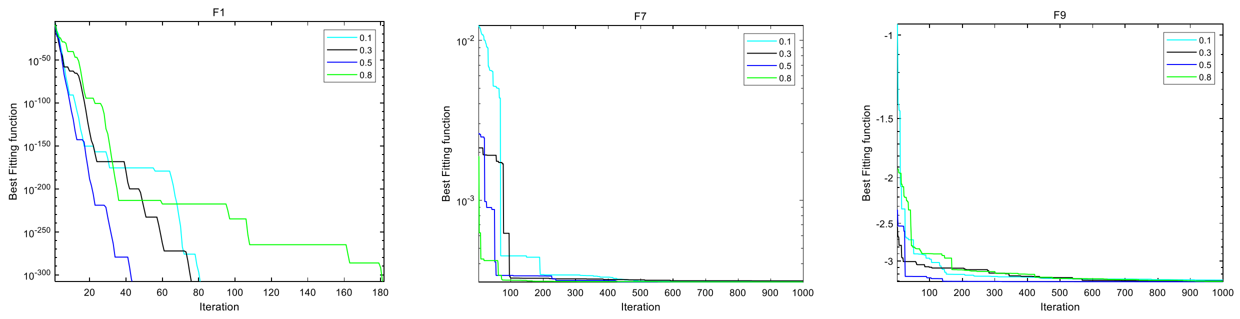

Figure 8. Sensitivity analysis of IBFO-A to parameter *pc*.

	F1		F7		F9	
	Score	Time	Score	Time	Score	Time
N						
30	0	0.26881	3.2398e−4	0.34955	− 3.2291	0.39857
50	0	0.28275	3.0818e−4	0.34564	− 3.2296	0.40963
80	0	0.33146	3.1231e−4	0.37591	− 3.3105	0.47671
100	0	0.35398	3.0753e−4	0.37827	− 3.3136	0.51034
P_{ed}						
0.4	0	0.43908	3.0939e−4	0.43906	− 2.4921	0.40801
0.6	0	0.4155	3.0869e−4	0.41504	− 2.6352	0.37525
0.8	0	0.34138	3.0749e−4	0.39222	− 3.2036	0.39457
0.95	0	0.36583	3.0749e−4	0.39482	− 3.3043	0.39887
pc						
0.1	0	0.33018	3.0956e−4	0.32449	− 3.2927	0.40793
0.3	0	0.34028	3.1217e−4	0.34316	− 3.3085	0.40045
0.5	0	0.34936	3.0761e−4	0.34173	− 3.3193	0.43215
0.8	0	0.35788	3.0766e−4	0.39245	− 3.3164	0.42945

Table 5. Parameter sensitivity comparison experiment.

thus improving the optimization ability of the flora. However, even if IBFO-A uses the improved reproductive activity, too high a *pc* probability may still lead to a decrease in bacterial diversity, putting the algorithm at risk of falling into a local optimal solution. Finally, according to the optimization results and running time analysis, IBFO-A algorithm is not sensitive to the change of hyperparameters within a reasonable range.

Comparison and analysis of twelve optimization algorithms in CEC2019 benchmark functions
The experimental design of algorithm optimization performance comparison is as follows: In addition to the eight optimization algorithms mentioned above, including IBFO-A, Two novel and improved optimization algorithms for PSO and BFO :AWPSO[25] and ChaoticBFO[26], as well as the two latest advanced optimization algorithms COA[63] and GO[64], a total of 12 optimization algorithms were compared in 10 sets of different types of advanced benchmark functions in CEC2019. We set experimental parameters uniformly for all optimization algorithms. The specific 10 groups of optimization convergence curves, scores and running schedules are shown in Fig. 9 and Table 6.

According to the experimental data in convergence diagram 9 and Table 6, compared with other 11 algorithms, IBFO-A has the best performance in four function scoring tests, and ASR ranks first, indicating that IBFO-A has good optimization performance. IBFO-A performs well in F1 and F10 test functions designed for single-objective real parameter optimization, demonstrating the IBFO-A algorithm's good performance in the global search for the best solution. It also performs well in the two high-dimensional test functions F2 and F3, This shows that the improved chemotactic activity and replication activity achieve a harmonious equilibrium between exploration and exploitation. It makes IBFO-A algorithm have better searching ability in test functions of different dimensions, and can be used to optimize DBN structure model. In addition, the basic algorithm BFO and another improved algorithm, ChaoticBFO, are also competitive in F1 and F2 compared with other test functions, but their performance is slightly inferior in F3 test functions, which may be because the random elimination-dispersal activity they use is difficult to escape the local optimal solution under high-dimensional functions. For the test function of fixed-dimensional multi-modal and multi-objective optimization, IBFO-A ranks among the best in F7-F9 and performs well in F4-F6, indicating that IBFO-A algorithm can be applied to multimodal and multi-objective optimization problems. In addition, the ATR of IBFO-A algorithm ranks 9th, indicating that the running time of IBFO-A algorithm has increased in complex optimization problems.

Two real-world engineering optimization problems
In this section, we use two different real-world engineering optimization problems to evaluate the model optimization capabilities of the IBFO-A algorithm, where each set of algorithms is run independently 50 times. Two kinds of engineering optimization problem parameter selection are shown in Table 7.

The first engineering optimization problem we chose was: Tension/compression spring design problem (TCSD)[65], TCSD is a continuously constrained problem such that the volume V of the coil spring is minimized under constant tension/compression load. The second engineering optimization problem we selected is Constrained truss optimization problem[66]. Three-bar truss is a common structural form in engineering, which is widely used in bridges, buildings, mechanical equipment and other fields. The optimization problem of structure design of three-bar truss is to get the best structure layout under certain constraints by adjusting the parameters such as the size, shape and connection mode of the bar. The running results of 12 optimization algorithms in TCSD and Three-bar truss engineering problems are shown in Tables 8 and 9

Two groups of experiments show that IBFO-A algorithm has improved optimization performance compared with the original BFO algorithm in finding the best objective function. In summary, IBFO-A algorithm has a good optimization ability in practical engineering applications.

Experimental results and analysis of algorithm convergence

The performance test experiment of the IBFO-D algorithm on network learning is conducted in two steps. In the first step, two types of data were randomly extracted from the alarm benchmark network:

(1) Three sets of non-temporal data samples, each containing 1000, 2000, and 3000 randomly selected sample points.
(2) Three sets of time series sample data. The time series data contains two time slices, and each set contains 1000, 2000, and 3000 randomly selected sample points.

As the optimization process is a random search, each iteration experiment is independently run 50 times to comprehensively evaluate and analyze the iterative convergence of the fitness values of the IBFO-D algorithm with respect to B_0 and $B_{\rightarrow}$. This analysis checks the stability of the learning network of the algorithm and whether it falls into its local optimal solution.

We chose to conduct convergence analysis experiments in the alarm network for two reasons:

(1) In the field of learning Bayesian network structures, the alarm network is widely recognized as the most popular benchmark.
(2) Compared with other network structures, the alarm network is more complex, and its performance on complex networks can better reflect the global search capability and stability of the IBFO-D algorithm.

Figures 10 and 11 illustrate the convergence of fitness scores during iterations, with the X-axis and Y-axis representing the number of iterations and the fitness score, namely the K2 score, respectively. In each generation,

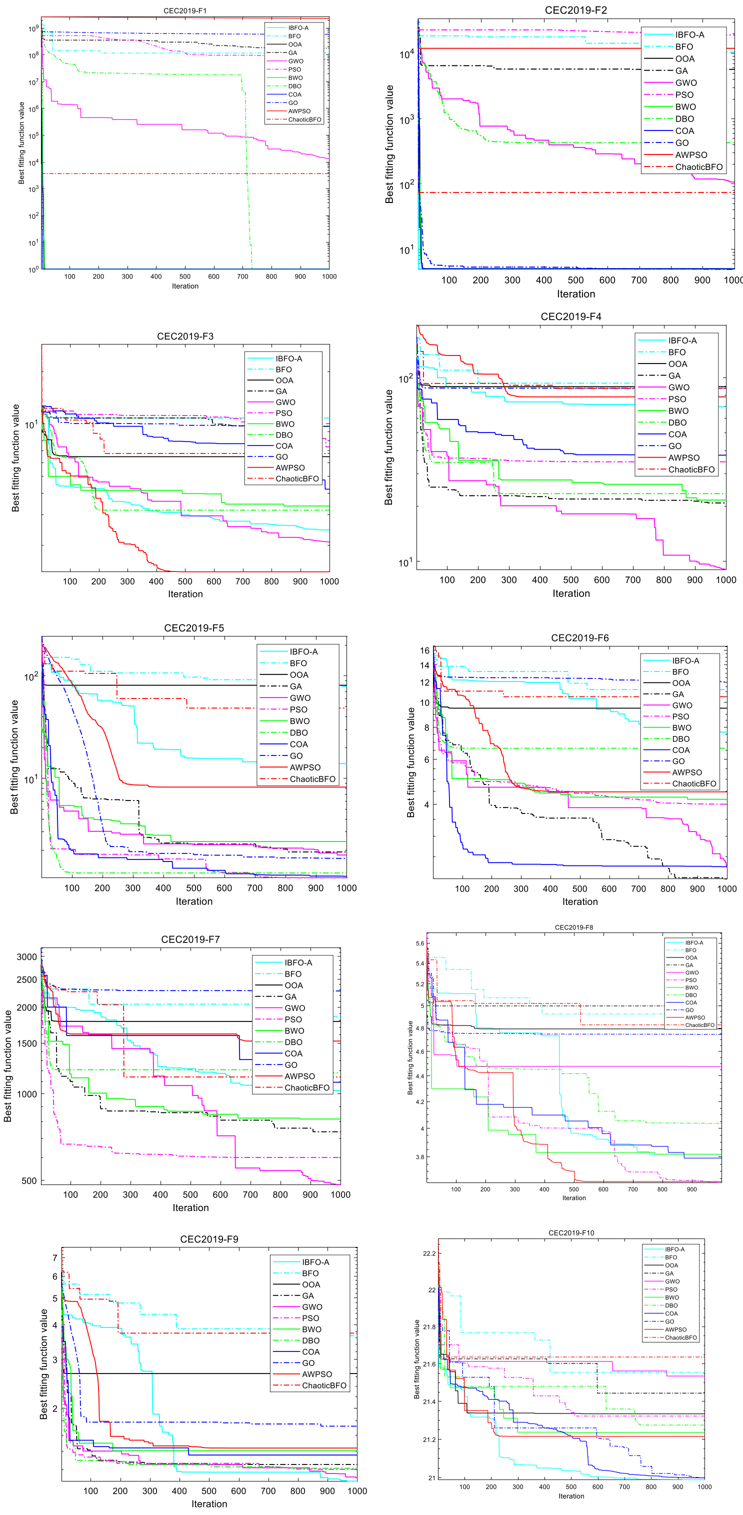

Figure 9. Comparison of convergence curves of 12 optimization algorithms.

the K2 score represents the average outcome of 50 independent runs of the algorithm. An analysis of the experimental findings shows that the algorithm reaches convergence at approximately 110 iterations for the three sets of temporal data, whereas for the other three sets of non-temporal data, convergence is achieved at approximately 70 iterations. This indicates that the IBFO-D algorithm can converge stably within a high fitness value in the temporal and non-temporal data without getting trapped in local optima due to its improved chemotaxis, reproduction, and elimination-dispersal strategies. Furthermore, this algorithm demonstrated a rapid convergence speed and good convergence accuracy.

	IBFO-A				BFO				OOA				GA				GWO				PSO			
	Score	Time	ASR|ATR		Score	Time	ASR|ATR		Score	Time	ASR|ATR		Score	Time	ASR|ATR		Score	Time	ASR|ATR		Score	Time	ASR|ATR	
F1	1	0.51471	1	9	1.1779e+8	0.13126	12	1	1	0.66938	10	11	1.7816e+8	0.57757	7	8	1.3322e+4	0.61903	3	6	6.5303e+7	1.0164	5	12
F2	4.9857	0.43262			1.0458e+4	0.20157			5	0.43205			5.6961e+3	0.51007			1.0655e+2	0.41483			1.9105e+4	0.59639		
F3	2.4498	0.31644			10.837	0.085054			6.4824	0.4427			9.6893	0.25438			2.0939	0.33925			7.3875	0.47824		
F4	69.4439	0.37462			78.9059	0.12656			89.3174	0.52289			20.7976	0.26875			9.0293	0.30214			34.8285	0.52836		
F5	13.8409	0.35037			79.3276	0.16411			81.1214	0.55487			1.9168	0.49689			1.7881	0.35106			1.0714	0.49645		
F6	7.6439	3.7762			11.216	0.72694			9.4888	4.6178			2.046	2.4695			2.2431	2.3497			4.0063	4.6507		
F7	1.0269e+3	0.41371			1.8581e+3	0.15723			1.7834e+3	0.55431			7.3754e+2	0.37282			4.8324e+2	0.58203			6.0044e+2	0.80284		
F8	3.8167	0.33895			4.8899	0.088168			4.7926	0.3799			4.9967	0.27517			4.4762	0.24203			3.6252	0.42283		
F9	1.0912	0.36772			3.6541	0.081567			2.6682	0.4817			1.2533	0.25649			1.1158	0.22874			1.2037	0.35686		
F10	20.9891	0.33267			21.555	0.16646			21.3348	0.46354			21.4439	0.56025			21.535	0.25705			21.3227	0.40833		

	BWO				DBO				COA				GO				AWPSO				ChaoticBFO			
	Score	Time	ASR|ATR		Score	Time	ASR|ATR		Score	Time	ASR|ATR		Score	Time	ASR|ATR		Score	Time	ASR|ATR		Score	Time	ASR|ATR	
F1	1	0.48294	4	10	1	0.45176	6	7	1	0.43022	2	4	5.9018e+8	0.4726	9	5	2.31e+9	0.36643	8	3	3.6731e+3	0.17772	11	2
F2	5	0.62992			4.2723e+2	0.44874			5	0.42175			4.998	0.60817			1.1964e+4	0.27733			73.933	0.17162		
F3	3.3701	0.40246			3.1877	0.27811			4.213	0.2029			9.7112	0.28002			1.4091	0.19957			6.7682	0.1404		
F4	21.5619	0.38112			23.3781	0.29205			37.838	0.37738			87.5607	0.28915			78.6063	0.4423			87.3588	0.21643		
F5	2.4201	0.42112			1.1945	0.30336			1.1084	0.33756			1.6568	0.34559			8.2245	0.20874			48.5569	0.13086		
F6	4.1681	2.9611			6.6196	2.4659			2.2831	3.8721			12.0047	2.4237			4.4743	2.3223			10.5205	0.76085		
F7	8.1706e+2	0.63241			1.182e+3	0.41399			1.095e+3	0.3335			2.2759e+3	0.3119			1.5242e+3	0.23214			1.1435e+3	0.13789		
F8	3.8176	0.50389			4.0369	0.36719			3.7907	0.33956			4.7452	0.35088			3.6329	0.21435			4.8279	0.17259		
F9	1.405	0.44			1.2145	0.44716			1.3554	0.4415			1.7225	0.28468			1.4362	0.19956			3.7324	0.19608		
F10	21.236	0.37332			21.2753	0.30392			20.999	0.25123			21.0001	0.29586			21.2158	0.21668			21.6367	0.13173		

Table 6. Comparison results of IBFO-A, BFO, OOA, GA, GWO, PSO, BWO, DBO, COA, GO, AWPSO, and ChaoticBFO on the CEC2019 benchmark functions.

Test function	Subject to	S
$\min f(x) = (x_3 + 2)x_2x_1^2$	$q_1(x) = 1 - \frac{x_2^3x_3}{71785x_1^4} \le 0$	$2 \le x_1 \le 15$ $0.25 \le x_2 \le 1.3$ $0.05 \le x_3 \le 2$
	$q_2(x) = \frac{4x_2^2 - x_1x_2}{1256(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0$	
	$q_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0$	
	$q_4(x) = \frac{x_2 + x_1}{1.5} - 1 \le 0$	
$\min f(x) = \left(2\sqrt{2}x_1 + x_2\right) \times l$	$g_1(x) = \frac{\sqrt{2}x_1 + x_2}{\left(\sqrt{2}x_1^2 + 2x_1x_2\right)}P - \sigma \le 0$	$0 \le x_i \le 1, i = 1, 2$
	$g_2(x) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \le 0$	
	$g_3(x) = \frac{1}{\sqrt{2}x_2 + x_1}P - \sigma \le 0$	

Table 7. Two kinds of engineering optimization problem parameter selection.

	IBFO-A	BFO	OOA	GA	GWO	PSO
Best	0.012667247	0.013558966	0.012668072	NAN	0.01266895	0.012666835
Worst	0.014586288	0.029029047	0.016834462	NAN	0.012729057	0.016726958
Average	0.013056487	0.021083917	0.013893121	NAN	0.012705313	0.013185834
Std	4.60e−04	3.59e−03	1.10e−03	NAN	1.95e−05	9.54e−04
	BWO	**DBO**	**COA**	**GO**	**AWPSO**	**ChaoticBFO**
Best	0.01273713	0.012674165	0.01267023	0.012669091	0.012666912	0.013769665
Worst	0.016191356	0.017773158	0.01351603	0.017044529	0.014294677	0.023413664
Average	0.013232493	0.013493124	0.012866426	0.013174405	0.012949943	0.013940312
Std	7.39e−04	1.61e−03	1.93e−04	9.22e−04	3.30e−04	4.64e−03

Table 8. The running results of 12 optimization algorithms in TCSD engineering problems.

	IBFO-A	BFO	OOA	GA	GWO	PSO
Best	263.8959581	263.9397436	263.9019434	NAN	263.8959079	NAN
Worst	264.0877488	264.8430515	270.2676865	NAN	263.9024999	NAN
Average	263.9072699	264.28099	264.9638793	NAN	263.8977088	NAN
Std	3.01e−02	2.63e−01	1.30e+00	NAN	1.53e−03	NAN
	BWO	**DBO**	**COA**	**GO**	**AWPSO**	**ChaoticBFO**
Best	263.9797514	263.8958434	263.895859	263.8958465	263.8958434	263.929491
Worst	264.9257365	263.8969615	263.8975473	263.8966012	263.8963019	265.1599811
Average	264.2141197	263.8959623	263.8962099	263.8959696	263.8958947	264.1313994
Std	1.81e−01	2.00e−04	3.29e−04	1.29e−04	8.55e−05	2.58e−01

Table 9. The running results of 12 optimization algorithms in Three-bar truss engineering problem.

Experimental results and analysis of the algorithm performance comparison

The second step involved the use of two dynamic benchmark networks as experimental models. The structural hamming distance (SHD)[67] was used to compare the IBFO-D algorithm with the DMMHC algorithm[9] and the GS algorithm[10] based on temporal information as a comprehensive evaluation metric. This comparison was conducted using three different sets of data samples. Each experiment was independently conducted 50 times to ensure thorough validation of the accuracy and efficiency of these algorithms.

The performance comparison results of the three different algorithms in the small-scale 2T-Asia network and the large-scale 2T-alarm network, across six different data samples, are presented in Tables 10, 11, 12, 13, 14 and 15. In these tables, $\mu \pm \sigma$ denotes the average value μ and standard deviation σ of the execution time (seconds) over 50 independent runs for each algorithm.

For the 2T-Asia network, the IBFO-D and DMMHC algorithms have the same optimal SHD when the sample size is 1000. However, the IBFO-D algorithm exhibits better stability and accuracy than the DMMHC algorithm with respect to the worst and average results. This is because the IBFO-D algorithm is based on global-search-based SI optimization, where the error of an individual agent does not affect the optimization outcome of the entire swarm. Furthermore, the reproductive activity in IBFO-D improves the information exchange capability among bacterial individuals, thereby enhancing the overall optimization performance of the bacterial population.

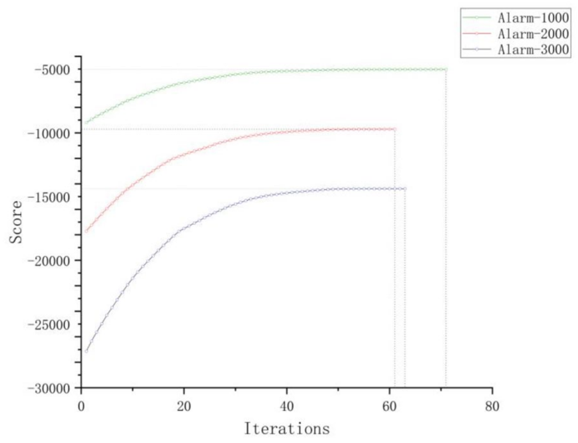

Figure 10. Experimental results of iterative convergence for three sets of non-temporal data samples.

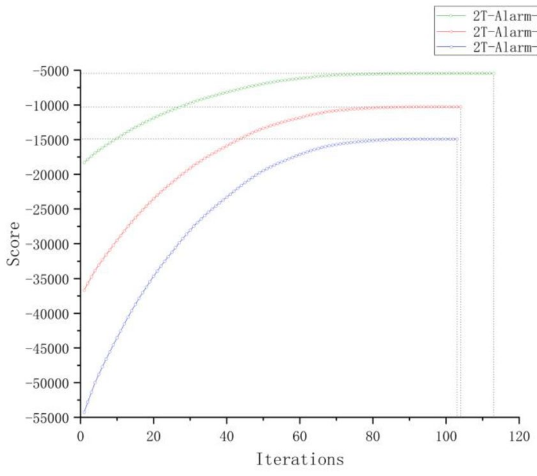

Figure 11. Experimental results of iterative convergence for three sets of 2T-BN temporal data samples.

(1) From the viewpoint of structural metrics, the differences in algorithm performance become more evident as the sample size increases and network complexity improves. The IBFO-D algorithm exhibits a clear advantage when the sample size is 3000, with a stable SHD of 1. Compared with IBFO-D, the DMMHC algorithm may have lower accuracy, but it generates networks that are relatively close to the true structure. However, the GS algorithm performs the worst, exhibiting the maximum structural variation in all scenarios. (2) The DMMHC algorithm is the fastest, closely followed by the IBFO-D algorithm, while the GS algorithm is the slowest when considering time metrics. These results can be attributed to the fact that global search typically requires more time than greedy local search. Furthermore, in SI algorithms, optimization and complete information exchange tasks are independently executed by individual agents during each iteration, leading to higher time costs. Notably, in the IBFO-D algorithm, significant time savings are achieved by omitting the grouping mechanism when searching for DBN structures in networks such as the small-scale Asia network.

For the 2T-alarm network: (1) When structural metrics are considered, the SHD values for all three algorithms are relatively large in the dataset with a sample size of 2000. This is because a small number of sample cases may

	DMMHC		GS		IBFO-D	
	SHD	TIME	SHD	TIME	SHD	TIME
Best	2	41.9	8	127.2	2	44.7
Worst	4	63.2	10	191.6	3	61.1
Average	3.5	49.6 ± 11.7	9.3	144.7 ± 15.9	2.2	53.4 ± 6.3

Table 10. Experimental results of the performance comparison of the three algorithms on 2T-Asia-1000.

	DMMHC		GS		IBFO-D	
	SHD	TIME	SHD	TIME	SHD	TIME
Best	2	89.8	7	210.7	1	97.6
Worst	3	117.2	10	284.2	2	114.2
Average	2.7	101.3 ± 15.8	7.9	253.3 ± 41.3	1.6	107.4 ± 8.4

Table 11. Experimental results of the performance comparison of the three algorithms on 2T-Asia-2000.

	DMMHC		GS		IBFO-D	
	SHD	TIME	SHD	TIME	SHD	TIME
Best	2	182.6	4	392.3	1	198.7
Worst	3	220.3	7	492.6	1	211.8
Average	2.2	196.3 ± 13.3	6.3	441.5 ± 47.1	1	205.2 ± 7.6

Table 12. Experimental results of the performance comparison of the three algorithms on 2T-Asia-3000.

	DMMHC		GS		IBFO-D	
	SHD	TIME	SHD	TIME	SHD	TIME
Best	31	523.7	42	712.6	25	645.5
Worst	40	1006.4	61	1326.2	29	752.7
Average	34.7	783.3 ± 256.7	53.3	1026.4 ± 302.4	27.4	697.1 ± 41.4

Table 13. Experimental results of the performance comparison of the three algorithms on 2T-alarm-2000.

	DMMHC		GS		IBFO-D	
	SHD	TIME	SHD	TIME	SHD	TIME
Best	24	1546.5	42	2164.1	13	1449.6
Worst	33	2152.8	58	3321.8	20	1813.1
Average	27.6	1798.8 ± 395.3	47.6	2713.5 ± 523.9	18.2	1590.1 ± 184.2

Table 14. Experimental results of the performance comparison of the three algorithms on 2T-alarm-5000.

	DMMHC		GS		IBFO-D	
	SHD	TIME	SHD	TIME	SHD	TIME
Best	20	2603.6	39	3765.3	9	2414.8
Worst	26	3319.4	45	5143.8	18	2926.7
Average	22.5	3120.5 ± 368.2	42.5	4246.1 ± 663.5	12.3	2753.7 ± 226.1

Table 15. Experimental results of the performance comparison of three algorithms on 2T-alarm-8000.

not fully reflect the network characteristics in complex networks, leading to challenges in the accurate learning of network structure by the algorithms. However, in all scenarios, the IBFO-D algorithm consistently exhibits smaller structural variations than the other two algorithms, with a worst-case SHD of 29. In the dataset with a sample size of 5000, a notable enhancement in algorithm performance was observed. On average, 91.8 of 110 edges were correctly identified by the IBFO-D algorithm, making it the best-performing algorithm among the three. In the dataset with a sample size of 8000, this advantage becomes even more pronounced. This enhancement is due to the improved chemotaxis and elimination-dispersal approach, which improves the global optimization capability of the IBFO-D algorithm and enables the escape from local optima, thereby facilitating the search for the global optimum structure. (2) Considering time metrics, learning networks in complex node sequences requires more time. Comparative experimental analysis revealed that the IBFO-D and DMMHC algorithms exhibit similar execution efficiencies on large-sample datasets, indicating that the improved chemotactic activity in the IBFO-D algorithm facilitates fast optimization for edge orientation, resulting in optimal time performance. There is one exception, where, in the 2T-alarm-2000 dataset, the DMMHC algorithm outperforms the IBFO-D algorithm in terms of the best runtime. This is due to the lack of a local optima escape mechanism in the DMMHC algorithm, resulting in it being trapped in local optima in complex networks with inadequate sample size. Among the three algorithms, the GS algorithm performs the worst, mainly because of the substantial amount of time spent searching the search space. Based on the aforementioned experimental analysis, it can be concluded that the IBFO-D algorithm is an effective approach for learning DBNs from data, as it can identify network structures with high scores and low structural variations.At the same time, it has high execution efficiency.

Conclusion

In this study, an IBFO-A was proposed using the logistics-sine chaotic mapping method to initialize the population and improve the chemotactic activity, reproductive activity, and elimination-dispersal activity of the bacteria by combining the OOA algorithm development stage, GA crossover idea, and adaptive method. To solve the problem of complex DBN learning structures due to the introduction of time information, an IBFO-D algorithm is proposed within the framework of the IBFO-A algorithm. In this algorithm, the fitness function and V-structure orientation rule were constructed, and simulation experiments were conducted on a series of reference functions, the 2T-Asia network and the 2T-Alarm network. The experimental results show that the initial population of the IBFOA algorithm using the chaotic mapping method can accelerate the iterative convergence speed, and the improved chemotactic activity and reproductive activity can improve the optimization ability of bacteria. Based on the adaptive elimination-dispersal activity, the algorithm can effectively prevent the local optimal to guide bacteria to find a better solution. The IBFO-D algorithm demonstrates stable convergence at higher fitness values in temporal and non-temporal data, and its performance is better than that of the other two algorithms. Future work will focus on applying the IBFO-D algorithm to learn higher-order dynamic Bayesian networks and time-varying dynamic Bayesian networks to reduce the complexity of their time computation. In addition, the improved BFO method will be combined with other meta-heuristic methods to further improve its ability to search for optimal datasets.

Data availability

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Received: 17 January 2024; Accepted: 3 April 2024
Published online: 09 April 2024

References

1. Kitson, N. K., Constantinou, A. C., Guo, Z., Liu, Y. & Chobtham, K. A survey of Bayesian Network structure learning. *Artif. Intell. Rev.* **56**, 1–94 (2023).
2. Shiguihara, P., Lopes, A. D. A. & Mauricio, D. Dynamic Bayesian network modeling, learning, and inference: A survey. *IEEE Access* **9**, 117639–117648 (2021).
3. Adabor, E. S. & Acquaah-Mensah, G. K. Restricted-derestricted dynamic Bayesian Network inference of transcriptional regulatory relationships among genes in cancer. *Comput. Biol. Chem.* **79**, 155–164 (2019).
4. Dang, S., Chaudhury, S., Lall, B. & Roy, P. K. The dynamic programming high-order dynamic Bayesian networks learning for identifying effective connectivity in human brain from fMRI. *J. Neurosci. Methods* **285**, 33–44 (2017).
5. Komurlu, C. *et al.* Active inference for dynamic Bayesian networks with an application to tissue engineering. *Knowl. Inf. Syst.* **50**, 917–943 (2017).
6. Qu, L. *et al.* Dynamic Bayesian network modeling based on structure prediction for gene regulatory network. *IEEE Access* **9**, 123616–123634 (2021).
7. Ghahramani, Z. Learning dynamic Bayesian networks. *International School on Neural Networks, Initiated by IIASS and EMFCSC*, 168–197 (1997).
8. Pérez-Ariza, C. B., Nicholson, A. E., Korb, K. B., Mascaro, S. & Hu, C. H. in *AI 2012: Advances in Artificial Intelligence: 25th Australasian Joint Conference, Sydney, Australia, December 4–7, 2012. Proceedings 25.* 902–913 (2012).
9. Trabelsi, G., Leray, P., Ben Ayed, M. & Alimi, A. M. in *Advances in Intelligent Data Analysis XII: 12th International Symposium, IDA 2013, London, UK, October 17–19, 2013. Proceedings 12.* 392–403 (2013).
10. Trabelsi, G. *New structure learning algorithms and evaluation methods for large dynamic Bayesian networks*, Université de Nantes; Ecole Nationale d'Ingénieurs de Sfax, (2013).
11. Gámez, J. A., Mateo, J. L. & Puerta, J. M. Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood. *Data Min. Knowl. Disc.* **22**, 106–148 (2011).
12. Srinivas, M. & Patnaik, L. M. Genetic algorithms: A survey. *Computer* **27**, 17–26 (1994).
13. Gámez, J. A. & Puerta, J. M. Searching for the best elimination sequence in Bayesian networks by using ant colony optimization. *Pattern Recogn. Lett.* **23**, 261–277 (2002).
14. Gheisari, S. & Meybodi, M. R. Bnc-pso: Structure learning of bayesian networks by particle swarm optimization. *Inf. Sci.* **348**, 272–289 (2016).
15. Sahin, F. & Devasia, A. *Distributed particle swarm optimization for structural Bayesian network learning*. (2007).
16. Mirjalili, S., Mirjalili, S. M. & Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **69**, 46–61 (2014).
17. Ji, J., Wei, H. & Liu, C. An artificial bee colony algorithm for learning Bayesian networks. *Soft Computing* **17**, 983–994 (2013).
18. Yang, X.-S. & He, X. Bat algorithm: Literature review and applications. *Int. J. Bio-inspired Comput.* **5**, 141–149 (2013).
19. Mirjalili, S. & Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **95**, 51–67 (2016).
20. Fister, I., Fister, I. Jr., Yang, X.-S. & Brest, J. A comprehensive review of firefly algorithms. *Swarm Evol. Comput.* **13**, 34–46 (2013).
21. Metzler, R., Chechkin, A. V., Gonchar, V. Y. & Klafter, J. Some fundamental aspects of Lévy flights. *Chaos Solitons Fractals* **34**, 129–142 (2007).
22. Mahdavi, S., Rahnamayan, S. & Deb, K. Opposition based learning: A literature review. *Swarm Evol. Comput.* **39**, 1–23 (2018).
23. Deng, W., Xu, J. & Zhao, H. An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. *IEEE access* **7**, 20281–20292 (2019).
24. Gao, Z.-M. & Zhao, J. An improved grey wolf optimization algorithm with variable weights. *Comput. Intell. Neurosci.* **2019** (2019).

25. Liu, W. *et al.* A novel sigmoid-function-based adaptive weighted particle swarm optimizer. *IEEE Trans. Cybern.* **51**, 1085–1093 (2019).
26. Zhang, Q. *et al.* Chaos enhanced bacterial foraging optimization for global optimization. *Ieee Access* **6**, 64905–64919 (2018).
27. Mou, J., Zhu, Q., Liu, Y. & Bai, Y. Multi-objective optimal thrust allocation strategy for automatic berthing of surface ships using adaptive non-dominated sorting genetic algorithm III. *Ocean Eng.* **299**, 117288 (2024).
28. Giri, P. K., De, S. S. & Dehuri, S. Adaptive neighbourhood for locally and globally tuned biogeography based optimization algorithm. *J. King Saud University-Comput. Inf. Sci.* **33**, 453–467 (2021).
29. Giri, P. K., De, S. S., Dehuri, S. & Cho, S. B. Biogeography based optimization for mining rules to assess credit risk. *Intell. Syst. Acc. Finance Manag.* **28**, 35–51 (2021).
30. Hemeida, A. *et al.* Implementation of nature-inspired optimization algorithms in some data mining tasks. *Ain Shams Eng. J.* **11**, 309–318 (2020).
31. Zhu, B. *et al.* A critical scenario search method for intelligent vehicle testing based on the social cognitive optimization algorithm. *IEEE Trans. Intell. Transp. Syst.* (2023).
32. Ashraf, N. M., Mostafa, R. R., Sakr, R. H. & Rashad, M. Optimizing hyperparameters of deep reinforcement learning for autonomous driving based on whale optimization algorithm. *Plos one* **16**, e0252754 (2021).
33. Zhang, C., Zhou, L. & Li, Y. Pareto optimal reconfiguration planning and distributed parallel motion control of mobile modular robots. *IEEE Trans. Ind. Electron.* (2023).
34. Khan, S. A. & Engelbrecht, A. P. A fuzzy particle swarm optimization algorithm for computer communication network topology design. *Appl. Intell.* **36**, 161–177 (2012).
35. Cao, B. *et al.* Multiobjective 3-D topology optimization of next-generation wireless data center network. *IEEE Trans. Ind. Inform.* **16**, 3597–3605 (2019).
36. Nonut, A. *et al.* A small fixed-wing UAV system identification using metaheuristics. *Cogent Eng.* **9**, 2114196 (2022).
37. Duan, H., Lei, Y., Xia, J., Deng, Y. & Shi, Y. Autonomous maneuver decision for unmanned aerial vehicle via improved pigeon-inspired optimization. *IEEE Trans. Aerosp. Electron. Syst.* (2022).
38. Li, G., Gao, X. & Di, R. in *2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS).* 245–250 (2014).
39. Xing-Chen, H., Zheng, Q., Lei, T. & Li-Ping, S. in *2007 IEEE Symposium on Artificial Life.*85–91 (2007).
40. Santos, F. P. & Maciel, C. D. in *5th ISSNIP-IEEE Biosignals and Biorobotics Conference (2014): Biosignals and Robotics for Better and Safer Living (BRC).* 1–6 (2014).
41. Daqi, J. & Wang, H. in *Proceedings of 2021 5th Chinese Conference on Swarm Intelligence and Cooperative Control.* 1315–1325 (2021).
42. Quesada, D., Bielza, C. & Larrañaga, P. in *International Conference on Hybrid Artificial Intelligence Systems.* 158–171 (2021).
43. Deng, Y.-J., Liu, H.-R., Wang, H.-Y. & Liu, B. Learning Dynamic Bayesian Networks structure based on a new hybrid K2-Bat learning algorithm. *J. Chin. Inst. Eng.* **44**, 41–52 (2021).
44. Passino, K. M. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Syst. Mag.* **22**, 52–67 (2002).
45. Guanglei, M. *et al.* A survey of Bayesian Network structure learning. *J. Beihang Univ.* https://doi.org/10.13700/j.bh.1001-5965.2023.0445 (2023).
46. Friedman, N., Murphy, K. & Russell, S. Learning the structure of dynamic probabilistic networks. https://arxiv.org/abs/1301.7374 (2013).
47. Serfozo, R. *Basics of Applied Stochastic Processes* (Springer, 2009).
48. Varol Altay, E. & Alatas, B. Bird swarm algorithms with chaotic mapping. *Artif. Intell. Rev.* **53**, 1373–1414 (2020).
49. Demir, F. B., Tuncer, T. & Kocamaz, A. F. A chaotic optimization method based on logistic-sine map for numerical function optimization. *Neural Comput. Appl.* **32**, 14227–14239 (2020).
50. Supriyono, H. & Tokhi, M. O. in *2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks.* 72–77 (2010).
51. Niu, B., Wang, J. & Wang, H. Bacterial-inspired algorithms for solving constrained optimization problems. *Neurocomputing* **148**, 54–62 (2015).
52. Naveen, S., Kumar, K. S. & Rajalakshmi, K. Distribution system reconfiguration for loss minimization using modified bacterial foraging optimization algorithm. *Int. J. Electr. Power Energy Syst.* **69**, 90–97 (2015).
53. Niu, B., Wang, H., Tan, L. & Li, L. in *2011 Seventh International Conference on Computational Intelligence and Security.* 76–80 (2011).
54. Niu, B., Fan, Y., Xiao, H. & Xue, B. Bacterial foraging based approaches to portfolio optimization with liquidity risk. *Neurocomputing* **98**, 90–100 (2012).
55. Chen, H., Zhu, Y., Hu, K. & Ma, L. Bacterial colony foraging algorithm: Combining chemotaxis, cell-to-cell communication, and self-adaptive strategy. *Inf. Sci.* **273**, 73–100 (2014).
56. Wang, L., Zhao, W., Tian, Y. & Pan, G. A bare bones bacterial foraging optimization algorithm. *Cogn. Syst. Res.* **52**, 301–311 (2018).
57. Zhao, W., Wang, L., Tian, Y. & Pan, G. A bare bones bacterial foraging optimization algorithm. *Cogn. Syst. Res.* **52**, 301–311 (2018).
58. Dehghani, M. & Trojovský, P. Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems. *Front. Mech. Eng.* **8**, 1126450 (2023).
59. De Jong, K. A. & Spears, W. M. A formal analysis of the role of multi-point crossover in genetic algorithms. *Ann. Math. Artif. Intell.* **5**, 1–26 (1992).
60. Suganthan, P. N. *et al.* Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *KanGAL Rep.* **2005005**, 2005 (2005).
61. Zhong, C., Li, G. & Meng, Z. Beluga whale optimization: A novel nature-inspired metaheuristic algorithm. *Knowl.-Based Syst.* **251**, 109215 (2022).
62. Xue, J. & Shen, B. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization. *J. Supercomput.* **79**, 7305–7336 (2023).
63. Jia, H., Rao, H., Wen, C. & Mirjalili, S. Crayfish optimization algorithm. *Artif. Intell. Rev.* **56**, 1919–1979 (2023).
64. Hamad, R. K. & Rashid, T. A. GOOSE algorithm: A powerful optimization tool for real-world engineering challenges and beyond. *Evol. Syst.*, 1–26 (2024).
65. Tzanetos, A. & Blondin, M. A qualitative systematic review of metaheuristics applied to tension/compression spring design problem: Current situation, recommendations, and research direction. *Eng. Appl. Artif. Intell.* **118**, 105521 (2023).
66. Panagant, N., Kumar, S., Tejani, G. G., Pholdee, N. & Bureerat, S. Many-objective meta-heuristic methods for solving constrained truss optimisation problems: A comparative analysis. *MethodsX* **10**, 102181 (2023).
67. Trabelsi, G., Leray, P., Ayed, M. B. & Alimi, A. M. in *2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO).* 1–6 (2013).

Acknowledgements
The authors sincerely thank the anonymous reviewers for their valuable and constructive comments.

Author contributions

T.T.L., C.G.W. and G.L.M. provided funding support and technical prestudy. G.L.M. and Z.L.C. devised novel algorithms. Z.L.C. completed the programming of the algorithm and the manuscript. B.W. and M.Z.Z. set up the experimental environment.

Funding

The authors would like to thank the Liaoning Revitalization Talents Program (no. XLYC2007144), Shenyang Natural Science Foundation (No. 22-315-6-09), National Defense Basic Scientific Research program of China (No. JH2023007) and National Natural Science Foundation of China (No. 62373261).

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-58806-0.

Correspondence and requests for materials should be addressed to Z.C.