Parameters identification of unknown delayed genetic regulatory networks by a switching particle swarm optimization algorithm[☆]

Yang Tang[*,a,b], Zidong Wang[a,c], Jian-an Fang[a]

[a]*College of Information Science and Technology, Donghua University, Shanghai 201620, P.R. China.*
[b]*Institute of Textiles and Clothing, Hong Kong Polytechnic University, Hong Kong, P.R. China.*
[c]*Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex, UB8 3PH, United Kingdom.*

Abstract

This paper presents a novel particle swarm optimization (PSO) algorithm based on Markov chains and competitive penalized method. Such an algorithm is developed to solve global optimization problems with applications in identifying unknown parameters of a class of genetic regulatory networks (GRNs). By using a evolutionary factor, a new switching PSO (SPSO) algorithm is first proposed and analyzed, where the velocity updating equation jumps from one mode to another according to a Markov chain, and acceleration coefficients are dependent on mode switching. Furthermore, a leader competitive penalized multi-learning approach (LCPMLA) is introduced to improve the global search ability and refine the convergent solutions. The LCPMLA can automatically choose search strategy using a learning and penalizing mechanism. The presented SPSO algorithm is compared with some well-known PSO algorithms in the experiments. It is shown that the SPSO algorithm has faster local convergence speed, higher accuracy and algorithm reliability, resulting in better balance between the global and local searching of the algorithm, and thus generating good performance. Finally, we utilize the presented SPSO algorithm to identify not only the unknown parameters but also the coupling topology and time-delay of a class of GRNs.

Key words: Genetic regulatory networks, Markov chain, switching particle swarm optimization (SPSO), parameter identification, time-delay

1. Introduction

Genetic regulatory networks (GRNs) have been extensively studied in the biological and biomedical sciences, and a large amount of results have been published over the past decade [1-17]. Recently, GRNs with time-delays in the form of differential equations have received particular research attention [2, 8-17]. It has been revealed that time-delays play

[☆]This research was partially supported by the National Natural Science Foundation of PR China (Grant No 60874113), the Research Fund for the Doctoral Program of Higher Education (Grant No 200802550007), the Key Creative Project of Shanghai Education Community (Grant No 09ZZ66), the Engineering and Physical Sciences Research Council EPSRC of the U.K. under Grant No. GR/S27658/01, an International Joint Project sponsored by the Royal Society of the U.K., and the Alexander von Humboldt Foundation of Germany.

[*]Corresponding author.

Email addresses: `tangtany@gmail.com` (Yang Tang), `Zidong.Wang@brunel.ac.uk` (Zidong Wang), `jafang@dhu.edu.cn` (Jian-an Fang) a very important role in dynamical modelling of genetic networks, and mathematical models without introducing the delay effects may even provide wrong predictions of the mRNA and protein concentrations [5, 6, 8]. On the other hand, when modeling complex networks [15, 16] including GRNs, the stochastic disturbances in real-world gene expression data is of great significance. The stochastic noise arises in gene expression in one of two ways, i.e., internal noise and external noise [4, 10, 12, 14]. Moreover, it is also shown that many biological networks are complex networks with small-world and scale-free properties [7]. It is remarkable that the filtering and stability analysis issues have been addressed in [8-17] for stochastic and/or delayed GRNs delayed GRNs have been addressed [8-17].

So far, GRNs have not yet been completely understood as how the genes are expressed in the right time and right place with the right amount. For the aim of identifying genes of interest and designing drugs, biologists are more interested in knowing the steady-state values of the actual network states, namely, the concentrations of the mRNA and protein. It is generally hard for biologists to obtain all the information due to various technical difficulties sometimes. Thus, the study of identifying the biological systems from the available data has comes in place. In [12], the authors dealt with filtering problem for estimating the states of stochastic delayed GRNs. In [17], an adaptive filtering problem of identifying unknown parameters of GRN was investigated. In [18, 19], PSO and GA are proposed to estimate the unknown parameters. Unfortunately, the identification of unknown delay has not been tackled yet due primarily to the mathematical difficulty, despite the fact that the time-delay plays a very important role in modelling a GRN as mentioned above.

In search of an algorithm candidate for identifying both the parameters and time-delays of GRNs, the particle swarm optimization (PSO) algorithm emerges as a competent one. PSO is a population-based heuristic global optimization technique, first introduced by Kennedy and Eberhart [20] and referred to as a swarm-intelligence technique. The PSO algorithm is inspired by the social behavior of animals such as bird flocking and fish schooling, etc. In this algorithm, the population is called a swarm, and the trajectory of each particle in the search space is adjusted by dynamically altering its velocity, according to its own flying experience and swarm experience in the search space. During the last decade, PSO algorithm has gained much attention and wide applications in different fields due to its effectiveness in performing difficult optimization issues, as well as simplicity of implementation and ability to fast converge to a reasonably good solution [20-34]. Unfortunately, PSO suffers from the premature convergence problem, which is particularly true in complex problems since the interacted information among particles in PSO is too simple to encourage a global search. Many efforts have been made to avoid the premature convergence [21-34] and, in this paper, we would like to try a novel approach to getting rid of the premature convergence problem - the Markovian switching and leader competitive penalized multi-learning approach.

Markov chain is a very useful and important tool in economic, decision, control system, physics, chemicals, etc [35, 36]. At each step, the Markov process may change its state from the current one to another, or remain in the same state, according to a certain probability distribution. The changes of state are called transitions, and the probabilities associated with various state-changes are called transition probabilities. As we know, the PSO may be stuck in local optimum since the diversity of PSO has declined quickly in the prophase of the search. Could we establish a Markov jumping (switching) velocity updating equation that considers quick convergence to the global optimum and also keep the swarm global search simultaneously by taking advantage of the current search information? Furthermore, a LCPMLA is introduced in this paper to improve the convergence speed and solution accuracy by learning and penalizing method designed in this paper. A series of experiments are performed on 8 benchmarks to validate the effectiveness of SPSO, compared with various PSOs. Finally, the proposed SPSO is used to identify the unknown parameters of delayed GRNs including the unknown coupling structure and time-delays.

The organization of this paper is listed as follows. In Section 2, the PSO and its variants are reviewed briefly. In Section 3, a Markovian switching approach and a LCPMLA are proposed in detail. Section 4 experimentally compares the proposed SPSO with some well-known existing PSOs on a set of benchmark optimizing problem. Analysis of parameters switching and LCPMLA is conducted in Section 5. In Section 6, parameter identification problem of GRNs including the topology and time-delays has been addressed using the proposed SPSO. In the end, conclusion remarks are provided in Section 7.

2. PSO algorithms

2.1. Traditional PSO algorithms

PSO is a population-based optimization approach. Each particle acts as a possible solution to the optimization task at hand. During each iteration, the accelerating direction of one particle determined by its own experience and the social cooperation in the swarm. This strategy means that if a particle discovers a promising new solution, all the other particles will move closer to it, exploring the region more thoroughly in the process.

In PSO, a swarm consists of S particles moving around in a D-dimensional search space. The position of the ith particle is denoted by a vector, $x_i(k) = (x_{i1}(k), x_{i2}(k), \cdots, x_{iD}(k))$, where $x_{in}(k) \in [x_{\min,n}, x_{\max,n}]$ $(1 \le n \le D)$ with $x_{\min,n}$ and $x_{\max,n}$ being lower and upper bounds for the nth dimension, respectively. During the search process, the particle successively adjusts its position toward the global optimum according to the two factors: the best position encountered by itself (*pbest*) denoted as $p_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$ and the best position in the whole swarm (*gbest*) denoted as $p_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$. The velocity of the ith particle at the kth iteration is represented by $v_i(k) = (v_{i1}(k), v_{i2}(k), \cdots, v_{iD}(k))$, and is limited to a maximum velocity $v_{i,\max} = (v_{i\,\max,1}, v_{i\,\max,2}, \cdots, v_{i\,\max,D})$. In this paper, the maximum velocity $V_{\max}$ is set to the 20% of the search range [31]. $r_{1,j}$ and $r_{2,j}$ are uniform random numbers samples from $U(0, 1)$. The parameters c_1 and c_2 are called acceleration coefficients, namely, *cognitive* and *social* parameters, respectively. The velocity and position of the particle at next iteration are updated according to the following equations:

$$\begin{aligned} &v_i(k+1) = w(k)v_i(k) + c_1 r_{1,j}(k)(p_i(k) - x_i(k)) + c_2 r_{2,j}(k)(p_g(k) - x_i(k)), \\ &x_i(k+1) = x_i(k) + v_i(k+1), \end{aligned} \tag{1}$$

where w is the so-called inertia weight.

2.2. Some Variants of PSO

PSO has attracted much attention since its simple concept and effectiveness. Many researchers have focused on improving its search performance using various methods.

One of the variants is that the inertia weight w was introduced [22, 23]. It is shown that a larger inertia weight tends to facilitate the global exploration and a smaller inertia weight achieves the local exploration to fine-tune the current search area [22]. In [22, 23], a linearly decreased inertia weight w over time (PSO-LDIW) is proposed, where w is given by the following equation:

$$w = (w_1 - w_2) \times \frac{\text{maxiter} - \text{iter}}{\text{maxiter}} + w_2, \tag{2}$$

where w_1 and w_2 are starting and final values of inertia weight, respectively; *iter* is the current iteration number and *maxiter* is the maximum number of the iteration. It is generally taken that starting value $w_1 = 0.9$ and final value $w_2 = 0.4$.

The constriction factor has been introduced into PSO for analyzing the convergence in [21]. It has been suggested that a constriction factor may help to ensure convergence. $w = 0.729$ and $c_1 = c_2 = 1.49$ are recommended in their work.

On the other hand, Ratnaweera *et al.* [24] have introduced PSO with time-varying acceleration coefficients (PSO-TVAC). The improvement has the same motivation and the similar techniques as the PSO-LDIW, in which, the cognitive coefficient c_1 is decreased linearly and the social coefficient c_2 is increased linearly over time as the follows:

$$\begin{aligned} c_1 &= (c_{1f} - c_{1i}) \times \frac{\text{maxiter} - \text{iter}}{\text{maxiter}} + c_{1i}, \\ c_2 &= (c_{2f} - c_{2i}) \times \frac{\text{maxiter} - \text{iter}}{\text{maxiter}} + c_{2i}, \end{aligned} \tag{3}$$

where c_{1i} and c_{2i} are the initial values of the acceleration coefficients c_1 and c_2; c_{1f} and c_{2f} are the final values of the acceleration coefficient c_1 and c_2, respectively. Usually, $c_{1i} = 2.5, c_{2i} = 0.5, c_{1f} = 0.5$ and $c_{2f} = 2.5$.

In addition, designing different types of topologies for improving the search performance of PSO is also an active research. In [27], a fully informed particle swarm (FIPS) algorithm was proposed, in which the information of the entire neighborhood is employed to guide the particles. Recently, some evolutionary operators such as selection [32], crossover [33] and mutation [34] have been introduced to the PSO. On the other hand, a comprehensive-learning PSO (CLPSO) [28] was presented. The CLPSO guide the particle using different neighbor's history best position to update its flying on different dimensions for improved performance in multimodal applications. More recently, an adaptive PSO was proposed in [25]. An evolutionary factor was introduced to identify four defined evolutionary states, including exploration, exploitation, convergence, and jumping out in each generation, which enables the automatic control of inertia weight and acceleration coefficients.

3. A novel switching PSO

In this section, a novel switching PSO is introduced for balancing the local search and global search, which can improve the search performance efficiently. A velocity updated equation with Markovian jumping parameters is proposed. Then, a leader competitive penalized multi-learning approach (LCPMLA) is presented to enhance the global and local search ability using an automatical control method.

3.1. A Switching Velocity Updated Equation

The population distribution properties described by evolutionary factor was introduced in [25]. This method can well depict the distance between the global best particle and other particles in the swarm. It was shown that the evolutionary factor can take the population distribution information into account. Here, we use the evolutionary factor to define four states.

The mean distance between the particles in the swarm can be written as follows

$$d_i = \frac{1}{S}\sum_{j=1}^{S}\sqrt{\sum_{k=1}^{D}(x_i^k - x_j^k)^2}, \tag{4}$$

where S and D are the the populations size and the dimensions, respectively.

d_g stands for the globally best particle among d_i. $d_{\max}$ and $d_{\min}$ represent the maximum and minimum distances in d_i, respectively. Thus, evolution factor E_f is defined by [25]

$$E_f = \frac{d_g - d_{\min}}{d_{\max} - d_{\min}}. \tag{5}$$

It has been revealed that the evolution factor E_f can describe the mean distance from the globally best particle to other particles, which would be minimal in the convergence state. In contrast, the mean distance would be maximal in the jumping-out state since the globally best particle tends to be away from the swarm. In [25], four sets S_1, S_2, S_3 and S_4 have been defined using E_f, which represent the states of convergence, exploration, exploitation and jumping out, respectively. Since the state transition would be nondeterministic, a fuzzy classification method is used to implement the classification. However, the classification method has three shortcomings:

(1) At each generation, the acceleration coefficients should be calculated.

(2) If the current global best particle is a local optimum away from swarm, the other particles will fly to the particle with a high speed. This will lead to the case that the swarm is stagnated in local optimum, which will reduce the convergence speed.

(3) The classification method is a little complicated to implement.

In the following, we introduce a mode-dependent velocity updating equation with Markovian switching parameters with the hope to overcome the above shortcomings and further improve the search abilities. For this purpose, let us introduce the Markov chain briefly.

Let $\xi(k)(k \geq 0)$ be a Markov chain taking values in a finite state space $\mathcal{S} = \{1, 2, \cdots, N\}$ with probability transition matrix $\Pi = (\pi_{ij})_{N\times N}$ given by

$$\mathcal{P} = \{\xi(k+1) = j|\xi(k) = i\} = \pi_{ij}, i, j = 1, 2, \cdots, N, \tag{6}$$

where $\pi_{ij} \geq 0 (i, j \in \mathcal{S})$ is the transition rate from i to j and $\sum_{j=1}^{N}\pi_{ij} = 1$. Here, $\xi(k) = 1$, $\xi(k) = 2$, $\xi(k) = 3$ and $\xi(k) = 4$ stand for the convergence, exploitation, explore and jumping out state, respectively.

Consider the following velocity and position updating equations with Markovian jumping parameters:

$$\begin{aligned} v_i(k+1) &= w(k)v_i(k) + c_1(\xi(k))r_{1,j}(k)(p_i(k) - x_i(k)) + c_2(\xi(k))r_{2,j}(k)(p_g(k) - x_i(k)), \\ x_i(k+1) &= x_i(k) + v_i(k+1), \end{aligned} \tag{7}$$

where $c_1(\xi(k))$ and $c_2(\xi(k))$ are the acceleration coefficients. All of them are mode-dependent on a Markov chain.

The formulation of classification and probability transition matrix are defined as follows

$$\xi(k) = \begin{cases} 1, & 0 \le E_f < 0.25, \\ 2, & 0.25 \le E_f < 0.5, \\ 3, & 0.5 \le E_f < 0.75, \\ 4, & 0.75 \le E_f \le 1, \end{cases} \tag{8}$$

$$\Pi = \begin{pmatrix} \varphi & 1-\varphi & 0 & 0 \\ \frac{1-\varphi}{2} & \varphi & \frac{1-\varphi}{2} & 0 \\ 0 & \frac{1-\varphi}{2} & \varphi & \frac{1-\varphi}{2} \\ 0 & 0 & 1-\varphi & \varphi \end{pmatrix}, \tag{9}$$

respectively. It is easy to see that the probability φ and $1-\varphi$ are used to control the state transition. At each generation, the Markov process may change its state from the current state to another, or remain in the same state, according to a certain probability distribution. This mechanism can predict the next state easily. Meanwhile, it can also describe the fuzzy transition between two near states, which can enhance the search diversity. It is also worth mentioning that the probability φ plays an important role in the evolution process. In this paper, φ is fixed as 0.9 in the evolutionary process for keeping the classification accuracy as well as the search diversity.

3.2. Control of the Inertia Weight

The inertia weight w is employed to balance the global and local search abilities. The evolutionary factor E_f shares some properties with the inertia weight since E_f is relatively large in jumping out state and is relatively small in the convergence state. Thus, the following mapping $w(E_f)$ is defined:

$$w(E_f) = 0.5E_f + 0.4 \in [0.4, 0.9], \qquad \forall E_f[0, 1]. \tag{10}$$

Note that inertia weight w is monotonic with E_f which makes w adapt to the search environment. In jumping-out state and exploration state, large w will lead to a global search. In contrast, small w will give rise to local search. In this paper, the initial value of w is set to 0.9.

3.3. The Selection of the Acceleration Coefficients

In this paper, the initial value of the acceleration coefficients $c_1(\xi(0))$ and $c_2(\xi(0))$ are both set to 2 and automatically controlled according to the evolutionary state. The techniques are developed as follows.

(1)In a jumping-out state, the globally best particle is jumping out of local optimum and flying to a better optimum. The other particle should fly to the globally best one as quick as possible. A large c_2 and a small c_1 helps to achieve the goal. In this sate, $c_1(4) = 1.8$ and $c_2(4) = 2.2$ are set to fly to the globally best particle rapidly.

(2) In an exploration state, it is of great importance to explore the optima. Therefore, a large c_1 and a relatively small c_2 will let the particle tend to explore individually. $c_1(3) = 2.2$ and $c_2(3) = 1.8$ are set to obtain this goal in this state.

(3) The exploitation state is likely to occur after an exploration state and before a convergence state. Hence, a relatively large c_1 and a relatively small c_2 will let the particles use local information while search the potential region. In this state, $c_1(2) = 2.1$ and $c_2(2) = 1.9$ are set to maintain the balance between the global search and local search capabilities.

(4) In the convergence state, the swarm clusters compactly and seems to find the globally optimal field. The particles should follow the current globally best particle with a large c_2 and a small c_1. However, as pointed in [25], this mechanism will cause the premature convergence. In order to avoid this case, we set $c_1(1) = c_2(1) = 2$ to maintain the search diversity of the swarm and get to the current global area simultaneously.

Different from the techniques proposed in [25], our method is simpler to implement in that we do not calculate the acceleration coefficients at each generation and the classification method is simple, and also can work well in the following experiments. On the other hand, the state jumping is according to the current state and the probability matrix. Note that the current state has a large probability to maintain its current state and has a small probability to switch to another state for avoiding the case that flying to local optima too fast, which ensures that keep the search diversity and fast convergence speed at the same time. The entire process of the Markovian jumping updated equation is illustrated in Fig. 1.

3.4. Leader competitive penalized multi-learning approach

In this subsection, a leader multi-learning penalized approach (LCPMLA) is developed to help the globally best particle to jump out of the local optimal areas and accelerate the convergence speed. This approach is embedded and controlled automatically in the evolutionary process, which is governed by activation probability variables. In [25], an elite learning method is designed to help the particle jump out of the local optima when the state is detected to be in a convergence state. This technique has three major limitations:

(1) The method is only embedded in the convergence state. However, if the globally best particle is stuck in local optimal area in other states, this will lead to the case that the particles fly to local optima, which is likely to cause premature problems. Thus, in other three states, the globally best particle still needs momentum to improve itself.

(2) Note that the elite learning method is designed for global search in [25], where the convergence speed is not ideal. We can also use local search learning technique to promote

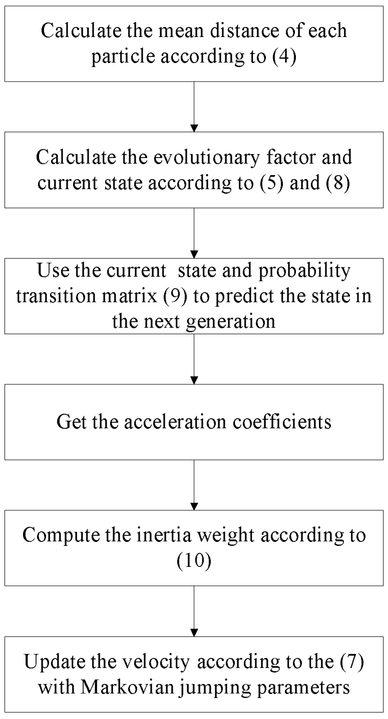

Calculate the mean distance of each particle according to (4)
Calculate the evolutionary factor and current state according to (5) and (8)
Use the current state and probability transition matrix (9) to predict the state in the next generation
Get the acceleration coefficients
Compute the inertia weight according to (10)
Update the velocity according to the (7) with Markovian jumping parameters

Figure 1: Flowchart of the parameter switching process.

the globally best particle to fly to the optimum more quickly.

(3) If the PSO is used to test the convergence speed, for example, the sphere function, the swarm is usually identified in convergence state in most generations. The elite learning approach may consume much computation time and fitness evaluations, which fails to find global optimum and reduce the convergence speed.

In the following, a LCPMLA using a non-homogeneous Markov chain is developed to improve the global search and local search capabilities at the same time.

Let $\gamma(k)(k \geq 0)$ be a *non-homogeneous* Markov chain taking values in a finite state space $\mathcal{S} = \{1, 2, \cdots, N\}$ with probability transition matrix $\Gamma^{(k)} = (\alpha_{ij}^{(k)})_{N\times N}$ given by

$$\mathcal{P} = \{\gamma(k+1) = j|\gamma(k) = i\} = \alpha_{ij}^{(k)}, i, j = 1, 2, \cdots, N, \tag{11}$$

where $\alpha_{ij}^{(k)} \geq 0(i, j \in \mathcal{S})$ is the transition rate from i to j and $\sum_{j=1}^{N} \alpha_{ij}^{(k)} = 1$. Here, it is worth mentioning that the probability transition matrix $\Gamma^{(k)}$ is time-varying in the search process.

The LCPMLA randomly select one dimension of globally best particle, which is represented by P_j for the jth dimension. Only one dimension is chosen in that the local optima are likely to have better solution in one dimension. Note that every dimension of the globally best particle has the same probability to be chosen. The leader multi-learning is performed by three search strategies as follows

$$P_j = P_j + ((\delta_{1,j} - \delta_{2,j}) * r_1 - \delta_{1,j}) * R_1, \tag{12}$$

$$P_j = P_j + ((\lambda_{1,j} - \lambda_{2,j}) * r_2 - \lambda_{1,j}) * R_2, \tag{13}$$

$$P_j = P_j + ((\eta_1 - \eta_2) * r_3 - \eta_1) * R_3, \tag{14}$$

where r_1, r_2 and r_3 are uniform random numbers sample from $U(0,1)$. $\delta_{1,j}$ and $\delta_{2,j}$ is the same as upper and lower bounds of the problem, i.e., $x_{\max,j}$ and $x_{\min,j}$. $\lambda_{1,j}$ and $\lambda_{2,j}$ are the upper and lower bounds of the each particle in the swarm in jth dimension at each generation. η_1 and η_2 are the maximal and minimal value of all dimensions of the globally best particle at each generation at each generation. R_1, R_2 and R_3 are the search radii of the learning method which are governed by a probability matrix $\Gamma^{(k)}$. It is easy to see that, (12) is used to increase global search ability. At the same time, (13) and (14) are exploited to improve local search ability of the swarm.

It is worth pointing out that (14) together with (13) are both necessary to promote the local search ability. The reason is listed as follows. The (13) is aim at updating jth dimension of the optimization problem. On the other hand, when we test the benchmarks, it has been found that the particles with bad fitness in the swarm are much further away from the global optimum than the globally best particle. Every dimension of the globally best particle is close to the corresponding dimension of optimum. If we only use (13), the search radius may be large leading to the global search instead of local search. It performs inefficiently to fly to the optimum since (13) will provide a much larger search scope than (14). The globally best particle tends to approach the optimum more accurately using both (13) and (14).

For choosing the R_1, R_2 and R_3, consider the following probability transition matrix

$$\Gamma^{(k)} = \begin{pmatrix} \alpha^{(k)} & 1-\alpha^{(k)} \\ \alpha^{(k)} & 1-\alpha^{(k)} \end{pmatrix}, \tag{15}$$

where $\alpha^{(k)}$ is a time-varying probability variable to select the radius. Since the three radii are chosen by the same probability transition matrix, thus we only take the selection of R_1 as the example. $\gamma(k) = 1$ denotes that the R_1 is taken as $\rho_1 = 1$. And $\gamma(k) = 2$ indicates that the R_1 is chosen as $\rho_2 = 0.1$. The pseudo code of this technique can be summarized as follows:

```
If ($\gamma(k)$=1)
$R_1 = 1$;
else
$R_1 = 0.1$.
```

It is suggested that $\alpha^{(k)}$ be linearly decreased along the generation number, which is given by

$$\alpha^{(k)} = (\alpha_1 - \alpha_2) \times \frac{\text{maxiter} - \text{iter}}{\text{maxiter}} + \alpha_2, \tag{16}$$

where α_1 and α_2 are the upper and lower bounds of $\alpha^{(k)}$. We set $\alpha_1 = 1$ and $\alpha_2 = 0$ in this paper, which indicates that the ρ_1 occurs frequently and ρ_2 seldom happens in the early stage. Conversely, in the latter stage, the ρ_1 seldom occurs and ρ_2 happens frequently, which refine the solution efficiently. This mechanism can make the swarm benefit the global search as well as local search. In LCPMLA, the new position will be accepted if the fitness is better than the current globally best position. Otherwise, the new position is abandoned.

On the other hand, in the early phase, the LCPMLA is activated with a relatively low probability since the search by velocity updating equation is efficient at the beginning. However, in the latter stage, the swarm always faces up to the situation that either converges to the global optimum or jump out of the local optima. Hence, the LCPMLA should be activated with a high probability. One problem should also be taken into account is that three search methods (12)-(14) are not always useful in the search process. For example, when using PSO optimizing the sphere function, the global search learning (12) should be seldom activated since the sphere function is usually used to test the convergence speed of the algorithms. However, the local search learning (13) and (14) should be activated more frequently. Hence, based on the above discussion, a competitive penalized learning method is proposed here.

Define Q, Q_1, Q_2 and Q_3 as LCPMLA activation probability variable, global search learning (12) activation probability variable, local search learning (13) activation probability variable, local search learning (14) activation probability variable, respectively. Q, Q_1, Q_2 and Q_3 belong to the range $[0, 1]$. We make use of Q to control the LCPMLA activating or not. Meanwhile, Q_1, Q_2 and Q_3 are used to control the activations of (12), (13) and (14), respectively. Q is updated as follows:

$$Q = \begin{cases} Q + \varepsilon, & \text{if one of the search strategy } i \\ & \text{ improve the globally best particle successfully,} \\ Q, & \text{if the search strategy } i \\ & \text{fails to improve the globally best particle,} \end{cases} \tag{17}$$

where ε is the learning rate, which is fixed as $\varepsilon = 0.05$ in this paper.

The activation probability variables Q_1, Q_2 and Q_3 are updated as follows:

$$Q_i = \begin{cases} Q_i + \beta, & \text{if the search strategy } i \\ & \text{ improve the globally best particle successfully,} \\ Q_i - \sigma, & \text{if the search strategy } i \\ & \text{fails to improve the globally best particle,} \end{cases} \tag{18}$$

where $i = 1, 2, 3$ denotes the search strategy (12), (13) and (14), respectively. β is the learning rate and σ is a penalizing rate. In practice, $\beta \gg \sigma$ at each iteration step. In this paper, we fix the $\beta = 0.5$ and $\sigma = 0.0005$.

Note that we do not add the penalizing rate in (17) since too variables are not easy to implement and control. Hence, we only use σ to penalize three search strategies. By using leader competitive penalized approach, the probability variables Q_1, Q_2 and Q_3 are automatically to adapt to appropriate values by gradually increase the probability variables or reduce them. In this paper, the controlling activation probability variables Q, Q_1, Q_2 and Q_3 are initialized to 0.5 and adaptively controlled according to the learning rate and penalizing rate.

Another important point is that the Q_1, Q_2 and Q_3 should be set the upper and lower bounds. If the magnitude of the updated $Q_i (i = 1, 2, 3)$ exceeds 1, then Q_i is assigned the value 1. Similarly, if Q_i decreases under 0.01, then Q_i is set to the value 0.01. This value can reduce inefficient search strategy as well as keep the search strategy i reoccurring with a small probability. The LCPMLA process is depicted in Fig.2.

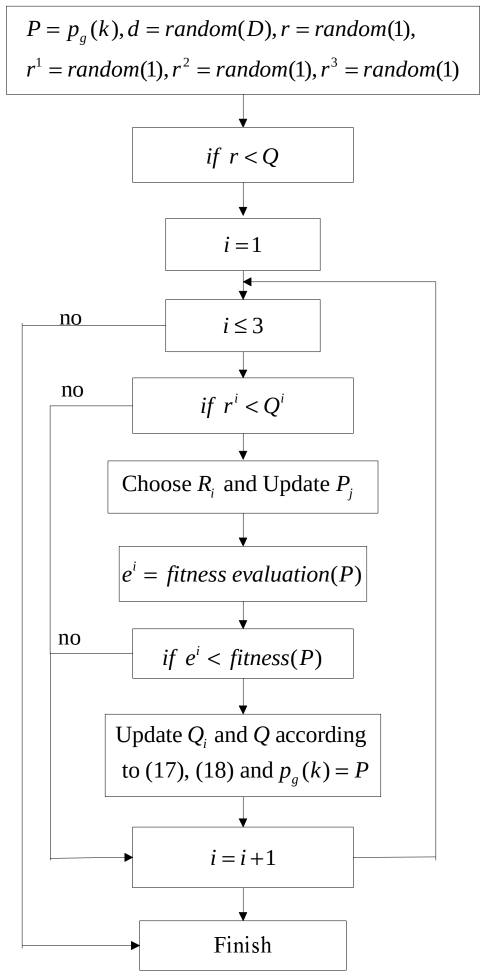

$P = p_g(k), d = random(D), r = random(1),$
$r^1 = random(1), r^2 = random(1), r^3 = random(1)$
$if\ r < Q$
$i = 1$
$i \le 3$
no
$if\ r^i < Q^i$
no
Choose R_i and Update P_j
e^i = fitness evaluation(P)
no
if e^i < fitness(P)
Update Q_i and Q according to (17), (18) and $p_g(k) = P$
$i = i + 1$
Finish

Figure 2: Flowchart of LCPMLA.

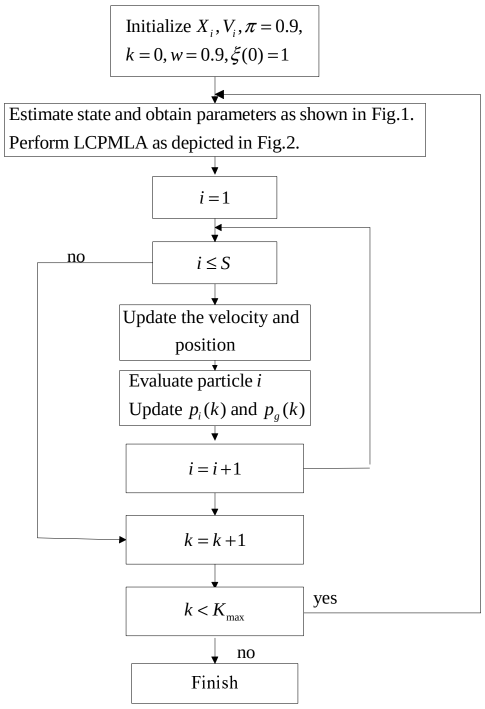
4. Experiments

In the experiments, some well-known benchmarks [37, 38] have been used to test the performances of SPSO with Markovian switching parameters. The experiments are carried out to validate the effectiveness of SPSO and compare the SPSO with other well-known PSOs to show its superiority.

4.1. Experiments setup

Eight benchmark functions are listed in Table I and (19)-(26) are used to test the performance of PSOs. All the functions are tested on 30 dimensions. The population size of all the PSOs are set to 20. $f_1(x)$, $f_2(x)$ and $f_3(x)$ are unimodal optimization problem. $f_1(x)$ and $f_2(x)$ are usually used to test the convergence rates of PSOs. $f_3(x)$ can be treated as a multimodal problem since it has a narrow valley from the perceived local optima to the global optimum. $f_4(x)$ to $f_8(x)$ are multimodal problems which are difficult to optimize.

$$\text{Sphere} : f_1(x) = \sum_{i=1}^{D} x_i^2, \tag{19}$$



Initialize $X_i, V_i, \pi = 0.9$, $k = 0, w = 0.9, \xi(0) = 1$
Estimate state and obtain parameters as shown in Fig.1. Perform LCPMLA as depicted in Fig.2.
$i = 1$
$i \le S$
no
Update the velocity and position
Evaluate particle i Update $p_i(k)$ and $p_g(k)$
$i = i + 1$
$k = k + 1$
$k < K_{\max}$
yes
no
Finish

Figure 3: Flowchart of the SPSO algorithm.

$$\text{Schwefel's P2.22} : f_2(x) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|, \tag{20}$$

$$\text{Rosenbrock} : f_3(x) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i)^2 + (x_i - 1)^2), \tag{21}$$

$$\text{Rastrigin} : f_4(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10), \tag{22}$$

$$\begin{aligned} \text{Noncontinuous Rastrigin} : \quad & f_5(x) = \sum_{i=1}^{D} (y_i^2 - 10\cos(2\pi y_i) + 10), \\ & \text{where} \quad y_i = \begin{cases} x_i, & |x_i| < 0.5, \\ \frac{round(2x_i)}{2}, & |x_i| \ge 0.5, \end{cases} \end{aligned} \tag{23}$$

$$\text{Ackley} : f_6(x) = -20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i} + 20 + e, \tag{24}$$

$$\text{Griewank} : f_7(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1, \tag{25}$$

Generalized Penalized :

$$f_8(x) = \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_D-1)^2\}$$
$$+\sum_{i=1}^{D} u(x_i,10,100,4),$$

$$\text{where} \quad y_i = (1+\frac{1}{4}(x_i+1)), u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i-a)^m, & x_i < -a. \end{cases} \quad (26)$$

Table 1: Benchmark configurations

Functions	Name	Dimension	Search Space	Minimum	Threshold
$f_1(x)$	Sphere	30	$[-100,100]^D$	0	0.01
$f_2(x)$	Schwelfel's P2.22	30	$[-10,10]^D$	0	0.01
$f_3(x)$	Rosenbrock	30	$[-10,10]^D$	0	100
$f_4(x)$	Rastrigin	30	$[-5.12,5.12]^D$	0	50
$f_5(x)$	Noncontinuous Rastrigin	30	$[-5.12,5.12]^D$	0	50
$f_6(x)$	Ackley	30	$[-32,32]$	0	0.01
$f_7(x)$	Griewank	30	$[-600,600]^D$	0	0.01
$f_8(x)$	Generalized Penalized	30	$[-50,50]^D$	0	0.01

Experiments are conducted to compare six PSO algorithms including the proposed SPSO on the 8 test problems with 30 dimensions. Five existing PSO algorithms are listed in detail in Table 2. The first PSO is PSO-LDIW [22, 23] with linearly decreasing inertia weight. PSO-TVAC [24] is a PSO with acceleration parameters and incorporating a self-organizing method, which is considered to be the second PSO. PSO-CK with constriction factor is proposed in [21]. CLPSO offers a comprehensive-learning strategy, which is used to yield better performance for multimodal functions [28]. APSO is a adaptive PSO that can adjust the acceleration coefficients and inertia weight automatically [25]. The parameters for these PSOs are provided in Table 2.

Table 2: PSO algorithms for comparison

Algorithm	Parameters	Reference
PSO-LDIW	$w : 0.9-0.4, c_1 = c_2 = 2$	[23]
PSO-TVAC	$w : 0.9-0.4, c_1 : 2.5-0.5, c_2 : 0.5-2.5$	[24]
PSO-CK	$w : 0.729, c_1 = c_2 = 2.05$	[21]
CLPSO	$w : 0.729, c = 1.49, m = 7$	[28]
APSO	Automatically chosen	[25]

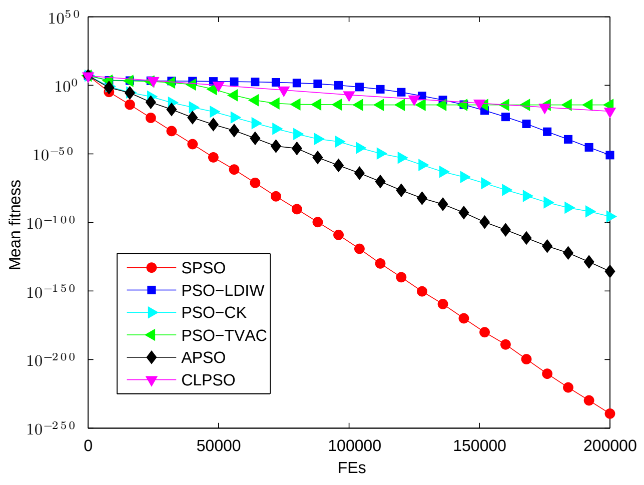


Figure 4: Performance of the algorithms for 30-dimensional $f_1(x)$ function when $S = 20$.

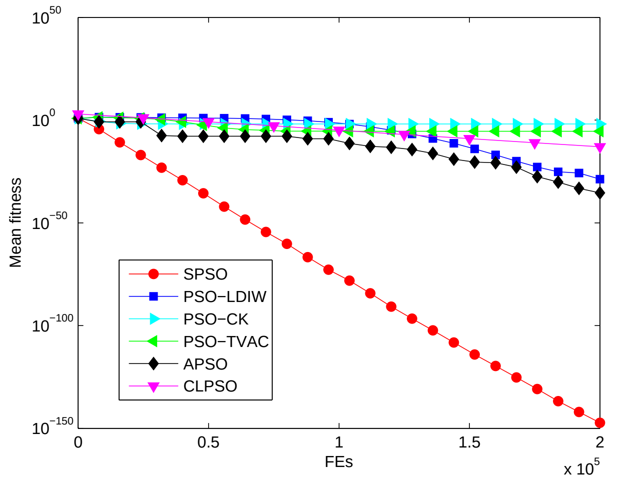


Figure 5: Performance of the algorithms for 30-dimensional $f_2(x)$ function when $S = 20$.

In all the experiments, the algorithm configuration of the SPSO is as follows. The inertia weight w is initialized to 0.9. The initial state $\xi(0)$ is set to 1 and the switching parameter φ is set to 0.9 in all the tests. Learning rates ε, β and penalizing rate σ are set to 0.05, 0.5 and 0.0005, respectively. Search radii ρ_1 and ρ_2 are chosen as 1 and 0.1, respectively.

In the tests, the population size is 20 for all the PSOs. Further, all the algorithms use the same number of 2×10^5 fitness evaluations (FEs) for each test function, as recommended in [38]. All the experiments are conducted on the same machine with a Core 2 2.26-GHz CPU, 2-GB memory, and Windows XP operating system. To eliminate random discrepancy, each algorithm will repeat 30 times independently.

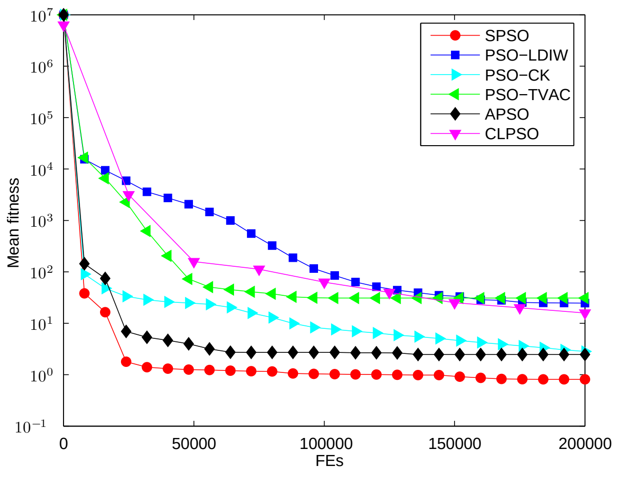


Figure 6: Performance of the algorithms for 30-dimensional $f_3(x)$ function when $S = 20$.

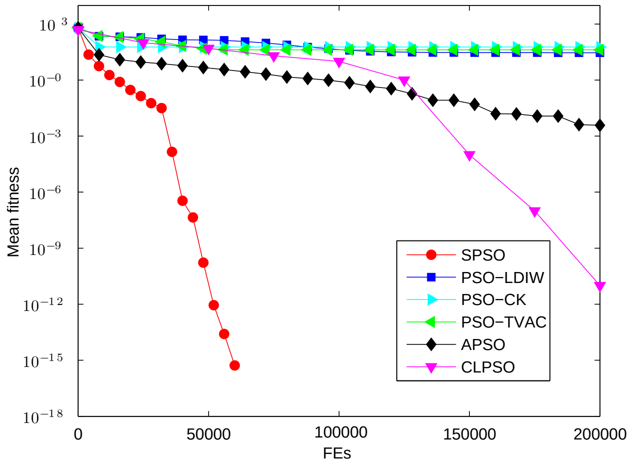


Figure 7: Performance of the algorithms for 30-dimensional $f_4(x)$ function when $S = 20$.

4.2. Comparisons on the solution accuracy

The results are listed in Table 3 in terms of the mean solutions, the best solution and standard deviation (Std. Dev.) of the solutions obtained in the 30 independent runs by each algorithm. The best result among those PSOs is indicated by Boldface in the table. Fig. 4-Fig. 11 depicts the comparisons in terms of convergence, mean solutions and evolution processes in solving 8 benchmark functions.

From the Table 3, Fig.4 and Fig.5, it is clearly that, the SPSO provides the best performance on the sphere and schwelfel functions, which are used to test the convergent rates. Table 3 and Fig. 6-Fig. 11 show the comparisons on the functions that difficult to optimize. SPSO achieves the global optimum on the optimization of complex functions f_4, f_5, f_6 and f_8. Furthermore, it offers the highest accuracy on functions f_3 and ranks second on f_7. Though CLPSO performs better than SPSO on f_7, its mean solutions

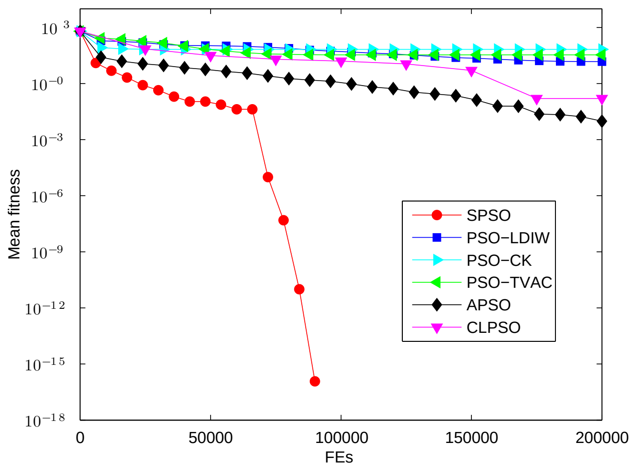


Figure 8: Performance of the algorithms for 30-dimensional $f_5(x)$ function when $S = 20$.

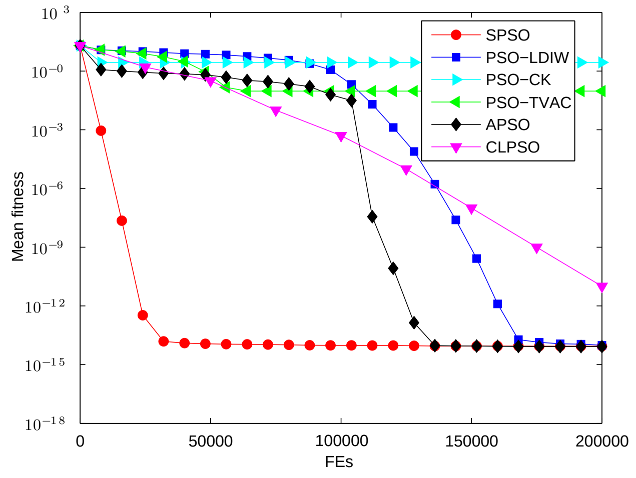


Figure 9: Performance of the algorithms for 30-dimensional $f_6(x)$ function when $S = 20$.

and best solution are worse than other results of the SPSO. It is also worth pointing out that the method proposed in this paper can help the PSO search the optimum as well as maintain a high convergence speed. The capability of avoiding local optima and finding global optimum of multimodal functions indicates that the superiority of SPSO.

4.3. Comparisons on convergent rate

The convergent rate for achieving the global optimum is a key point for measuring the algorithm performance. Note that in solving real-world optimization problems, the ”FE” overwhelms the algorithm overhead. Usually, the FE accounts for the most time as the PSO is highly computation efficient. Hence, the computation times of these algorithms are not compared here. Table 4 shows that SPSO needs much less FEs to achieve the acceptable solution, which reveals that SPSO has a much higher convergent rate than other algorithms. The SPSO uses the least number of FEs to achieve the acceptable

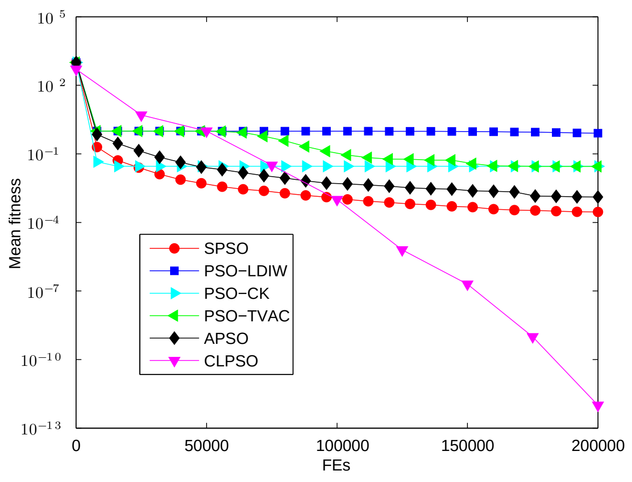


Figure 10: Performance of the algorithms for 30-dimensional $f_7(x)$ function when $S = 20$.

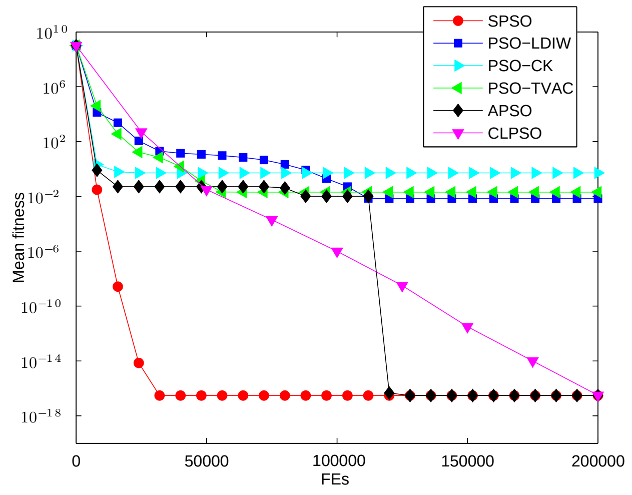


Figure 11: Performance of the algorithms for 30-dimensional $f_8(x)$ function when $S = 20$.

solutions on f_1, f_2, f_3, f_5, f_6 and f_8. Although PSO-CK needs less FEs on f_4 and f_7 than SPSO, the successful ratio of PSO-CK in 30 runs on f_4 and f_7 are much poorer than SPSO.

4.4. Comparisons on successful ratio

Table 4 also shows that SPSO yields a highest ratio for achieving acceptable solutions in 30 runs, together with APSO and CLPSO. Though APSO and CLPSO reach the acceptable solutions with the same successful ratio with SPSO, the convergence speed and mean solutions are worse than SPSO on all the functions except f_7. According to the "no free lunch" theorem [39], "any elevated performance over one class of problems is offset by performance over another class". Hence, one algorithm cannot perform better than all

Table 3: Search result comparisons among six PSOs on eight test functions

		PSO-LDIW	PSO-TVAC	PSO-CK	CLPSO	APSO	SPSO
f_1	Mean	1.31×10^{-51}	1.65×10^{-18}	9.88×10^{-91}	1.89×10^{-19}	4.23×10^{-132}	$\mathbf{9.5}\times10^{-241}$
	Best Value	2.20×10^{-56}	3.27×10^{-31}	2.51×10^{-122}	2.23×10^{-23}	3.94×10^{-148}	$\mathbf{2.7}\times10^{-254}$
	Std. Dev.	6.49×10^{-51}	6.42×10^{-18}	5.41×10^{-90}	1.49×10^{-19}	5.67×10^{-145}	**0**
f_2	Mean	5.67×10^{-29}	3.49×10^{-6}	0.0123	1.01×10^{-13}	1.89×10^{-37}	$\mathbf{4.9}\times10^{-148}$
	Best Value	0.10×10^{-36}	1.28×10^{-10}	1.10×10^{-19}	4.38×10^{-15}	2.3×10^{-45}	$\mathbf{1.0}\times10^{-150}$
	Std. Dev.	3.82×10^{-33}	3.49×10^{-8}	3.27×10^{-7}	6.51×10^{-14}	7.1×10^{-40}	$\mathbf{1.7}\times10^{-149}$
f_3	Mean	29.51	32.85	2.82	11	2.1	**0.8**
	Best Value	3.98	2.51	0.13	0.25	0.0006	$\mathbf{9.3}\times10^{-5}$
	Std. Dev.	37.10	24.97	3.95	14.5	1.1	**1.5**
f_4	Mean	25.9	48.9	58.2	2.57×10^{-11}	7.23×10^{-1}	**0**
	Best Value	13.9	24.8	21.7	6.93×10^{-12}	2.61×10^{-14}	**0**
	Std. Dev.	26.1	10.6	15.4	6.64×10^{-11}	0.68	**0**
f_5	Mean	14.8	34.1	67.4	0.16	0.0031	**0**
	Best Value	7.9	16.2	42.9	0.03	1.23×10^{-15}	**0**
	Std. Dev.	14.3	15.7	18.3	0.32	0.004	**0**
f_6	Mean	9.7×10^{-15}	0.09	2.7	2.0×10^{-12}	9.5×10^{-15}	$\mathbf{8.45}\times10^{-15}$
	Best Value	7.7×10^{-15}	2.8×10^{-13}	0.9	1.2×10^{-12}	7.7×10^{-15}	$\mathbf{4.15}\times10^{-15}$
	Std. Dev.	7.4×10^{-14}	0.3631	1.2	9.2×10^{-13}	2.5×10^{-15}	$\mathbf{2.45}\times10^{-15}$
f_7	Mean	0.77	0.02	0.0557	2.0×10^{-12}	0.0008	**0.0002**
	Best Value	0.08	1.6×10^{-8}	4.3×10^{-11}	4.3×10^{-16}	0.0001	$\mathbf{3.3}\times10^{-16}$
	Std. Dev.	0.17	0.12	0.0615	9.2×10^{-13}	0.001	**0.0002**
f_8	Mean	0.007	0.02	0.5	$\mathbf{3.0}\times10^{-17}$	$\mathbf{3.0}\times10^{-17}$	$\mathbf{3.0}\times10^{-17}$
	Best Value	3.0×10^{-17}	3.0×10^{-17}	3.0×10^{-17}	$\mathbf{3.0}\times10^{-17}$	$\mathbf{3.0}\times10^{-17}$	$\mathbf{3.0}\times10^{-17}$
	Std. Dev.	0.026	0.05	0.9	$\mathbf{1.41}\times10^{-26}$	$\mathbf{1.41}\times10^{-26}$	$\mathbf{1.41}\times10^{-26}$

Table 4: Convergence speed and algorithm reliability comparisons; '-' representing no runs reached an acceptable solution

		PSO-LDIW	PSO-TVAC	PSO-CK	CLPSO	APSO	SPSO
f_1	Mean FEs	106534	45339	9190	72081	7978	**4848**
	Ratio(%)	**100**	**100**	**100**	**100**	**100**	**100**
f_2	Mean FEs	103910	45741	14437	66525	24254	**4723**
	Ratio(%)	**100**	**100**	**100**	**100**	**100**	**100**
f_3	Mean FEs	97327	46614	5518	74815	4756	**4335**
	Ratio(%)	**100**	**100**	**100**	**100**	**100**	**100**
f_4	Mean FEs	92437	35056	**1393**	53416	3583	2076
	Ratio(%)	**100**	70	4 0	**100**	**100**	**100**
f_5	Mean FEs	101656	47491	4153	47440	3160	**1648**
	Ratio(%)	**100**	83.3	20	**100**	**100**	**100**
f_6	Mean FEs	110427	54642	-	47740	40209	**5608**
	Ratio(%)	**100**	96.7	0	**100**	**100**	**100**
f_7	Mean FEs	-	61266	**1458**	81422	72629	32405
	Ratio(%)	0	67.7	16.7	**100**	**100**	**100**
f_8	Mean FEs	77340	29626	8743	59160	27773	**5208**
	Ratio(%)	80	86.7	50	**100**	**100**	**100**
	Mean Reliability	85	88.17	52.15	**100**	**100**	**100**

the others on every problem.

In summery, the SPSO outperforms best on both unimodal and multimodal functions. The SPSO processes capabilities of fast convergence, highest successful ratio, least FEs and best search accuracy among these PSOs. The performance arises from the Markovian switching and LCPMLA embedded in the SPSOs.

5. Analysis of parameters switching and leader competitive penalized multi-learning approach

In this section, we carry out the experiments to show the sensitivities of penalized coefficient σ, the switching probability φ. Meanwhile, two techniques, i.e., parameters switching and leader competitive penalized multi-learning approach are also used to test the effects of them on the search performance of SPSO.

5.1. Advantages of parameters switching and leader competitive penalized multi-learning approach

The performance of SPSO without parameters switching and leader competitive penalized multi-learning approach is tested. Results of 30 independent runs are revealed in Table 5.

It is obvious from the results that with both parameters switching and leader learning techniques, SPSO outperforms other variants of SPSO. SPSO can not only offers a highest accuracy on unimodal functions, but also delivers a good global search performance on multimodal functions. Moreover, with only leader multi-learning technique, SPSO can still performs well on multimodal functions, which can efficiently help the globally best particle to jump out of local optima. However, SPSO with only leader learning suffers from low accuracy when solving unimodal functions.

On the other hand, the SPSO with only parameters switching and without leader learning can deliver a good performance on unimodal function. However, it cannot solve multimodal functions well. It is also worth mentioning that, SPSO with parameters switching, combining with LCPMLA can provide a much better performance on unimodal functions than that of SPSO with only parameters switching. It has been shown from Table 5 that LCPMLA can enhance the local search capability, when it is embedded in PSO with parameter switching. However, PSO with LCPMLA alone and without parameter switching only offers a good performance on multimodal functions, as discussed above.

To summarize, the full SPSO is the most powerful for any tested functions. The results verify that parameters switching can accelerate the convergence and LCPMLA can speed up the convergence of the algorithm as well as help the swarm to have a better global search ability.

5.2. Sensitive analysis of penalizing rate

In this subsection, the effect of penalizing rate σ is investigated here. The learning rate ε and β are fixed here for showing the effect of σ evidently. An appropriate σ can enhance global and local search capabilities and reduce inefficient search strategy resulting

Table 5: Advantages of parameter switching and LCPMLA

	Algorithms	SPSO with both switching and learning	SPSO with switching	SPSO with learning	SPSO without either (PSO-LDIW)
f_1	Average	$\mathbf{2.7}\times10^{-241}$	8.1×10^{-149}	7.5×10^{-96}	1.3×10^{-51}
	Std. Dev.	**0**	2.2×10^{-148}	3.9×10^{-95}	6.5×10^{-51}
f_2	Average	$\mathbf{4.9}\times10^{-148}$	2.2×10^{-61}	2.4×10^{-34}	5.7×10^{-29}
	Std. Dev.	$\mathbf{1.7}\times10^{-149}$	6.7×10^{-65}	3.5×10^{-35}	3.8×10^{-33}
f_4	Average	**0**	54.0	0	25.9
	Mean FEs	**0**	15.1	0	26.1
f_5	Average	**0**	20.2	0	14.8
	Mean FEs	**0**	9.8	0	14.3
f_8	Average	$\mathbf{3.0}\times10^{-17}$	0.49	2.4×10^{-5}	0.007
	Mean FEs	$\mathbf{1.4}\times10^{-17}$	0.57	3.1×10^{-5}	0.026

in saving the consumption of FEs and computing time. The results of mean values and standard deviations of the solutions are shown in Table 6. σ is fixed to 0.0001, 0.0005 and 0.001 in the investigation, respectively.

From the results, it can be found that $\sigma = 0.0005$ reveals the best performance. The reason can be listed as follows. A large σ gives rise to the case that leader multi-learning vanishes rapidly, while a small σ leads to the case that inefficient search strategy exists for a number of FEs. Hence, a small σ cannot solve the problem with high accuracy either. Therefore, in this paper, $\sigma = 0.0005$ is adopted in our paper, which can balance the efficient search and the search capability.

Table 6: Effects of the penalized rate on search accuracy

	σ	0.0001	0.0005	0.001
f_1	Average	4.2×10^{-234}	$\mathbf{9.5}\times10^{-241}$	3.2×10^{-240}
	Mean FEs	**0**	**0**	**0**
f_2	Average	7.6×10^{-145}	$\mathbf{4.9}\times10^{-148}$	3.2×10^{-143}
	Mean FEs	6.2×10^{-146}	$\mathbf{1.7}\times10^{-149}$	9.1×10^{-145}
f_4	Average	**0**	**0**	0.03
	Mean FEs	**0**	**0**	0.18
f_5	Average	**0**	**0**	**0**
	Mean FEs	**0**	**0**	**0**
f_8	Average	$\mathbf{3.0}\times10^{-17}$	$\mathbf{3.0}\times10^{-17}$	0.02
	Mean FEs	$\mathbf{1.4}\times10^{-26}$	$\mathbf{1.4}\times10^{-26}$	0.06

5.3. Sensitive analysis of switching probability

In order to measure the sensitivity of switching probability φ, five strategies for setting the value of φ to $0.8, 0.85, 0.9, 0.95$ and 1, respectively. The mean results and FEs are presented in Table 7.

It can be seen form the results that $\varphi = 0.9$ yields the a little better convergence speed and higher accuracy than other cases. The reason is that $\varphi = 0.9$ can balance the good classification result of $\xi(k)$ and diversity of search states.

Table 7: Effects of the switching parameter on search performance

	φ	0.8	0.85	0.9	0.95	1
f_1	Average	6.48×10^{-239}	$\mathbf{1.97}\times10^{-241}$	9.5×10^{-241}	1.6×10^{-238}	2.1×10^{-239}
	Mean FEs	4890	4944	**4848**	4893	5108
f_3	Average	4.7	3.9	**0.8**	1.8	3.1
	Mean FEs	5393	5639	**4723**	5399	5988
f_6	Average	9.1×10^{-15}	9.35×10^{-15}	$\mathbf{8.45}\times10^{-15}$	8.99×10^{-15}	9.13×10^{-15}
	Mean FEs	5643	5793	**5608**	5792	5653
f_7	Average	0.00024	0.00027	0.00021	**0.00020**	0.00033
	Mean FEs	36268	33763	**32405**	33138	33010

6. Parameter identification of unknown delayed stochastic GRN

In this section, we will utilize the proposed novel SPSO to identify the unknown parameters of the GRN including the unknown global coupling matrix and time-delays.

6.1. The GRN model

Consider the delayed GRN model in compact form

$$\begin{aligned}\frac{dm(t)}{dt} &= -Am(t) + Wf(p(t-\tau_1)) + L,\\ \frac{dp(t)}{dt} &= -Cp(t) + Dm(t-\tau_2).\end{aligned} \tag{27}$$

where $m(t) = [m_1(t), m_2(t), \cdots, m_n(t)]^T, p(t) = [p_1(t), p_2(t), \cdots, p_n(t)]^T$ are concentrations of mRNA and protein of the ith node at time t, $f(p(t-\tau_1)) = [f_1(p_1(t-\tau_1)), f_2(p_2(t-\tau_1)), \cdots, f_n(p_n(t-\tau_1))]^T$ with $f_j(p_j(t))$ as a monotonic increasing function of the form $f_j(p_j(t)) = (p_j(t)/\beta_j)^{H_j}/(1 + (p_j(t)/\beta_j)^{H_j})$, where H_j are called the Hill coefficients, β_j are positive constants. $A = diag(a_1, a_2, \cdots, a_n) > 0$ and $C = diag(c_1, c_2, \cdots, c_n) > 0$ are the degradation rates of the mRNA and protein, respectively; $D = diag(d_1, d_2, \cdots, d_n)$ is the translation rate, the constants τ_1, τ_2 denote respectively, the translation delay and the feedback regulation delay. $L = diag(l_1, l_2, \cdots, l_n)$, where $l_i = \sum_{j\in I_i} \alpha_{ij}$ in which I_i is the set of all the repressors of gene i, with $W = (w_{ij}) \in \mathbb{R}^{n\times n}$ defined as

$$w_{ij} = \begin{cases} \alpha_{ij}, & \text{if transcription factor } j \text{ is an activator of gene } i,\\ 0, & \text{if there is no link from node } j \text{ to } i,\\ -\alpha_{ij}, & \text{if transcription factor } j \text{ is a repressor of gene } i.\end{cases} \tag{28}$$

When modeling a realistic GRNs, molecular noise has been revealed to play an important role in biological functions since noise is ubiquitous in reactions of transcription, translation, and translocation processes, and also owing to external fluctuations. We consider the following networks of N coupled genetic oscillators with stochastic perturbations based on Eq. (1) and [10]:

$$
\begin{aligned}
dm_i(t) &= [-Am_i(t) + Wf(p_i(t-\tau_1)) + L + \sum_{j=1}^{N} G_{ij}\Gamma m_i(t)]dt + \delta_i(t)d\omega(t), \\
dp_i(t) &= [-Cp_i(t) + Dm_i(t-\tau_2) + \sum_{j=1}^{N} G_{ij}\Gamma p_i(t)]dt + \phi_i(t)d\nu(t), \\
i &= 1, 2, \cdots, N.
\end{aligned} \tag{29}
$$

where $\Gamma \in \mathbb{R}^{n\times n}$ defines the coupling between two genetic oscillators. $G = (G_{ij})_{N\times N}$ is the coupling matrix of the network. If there is a link from oscillator j to oscillator $i(j \neq i)$, then G_{ij} is a constant denoting the coupling strength of this link; otherwise, $G_{ij} = 0$; $G_{ii} = \sum_{j=1,j\neq i}^{N} G_{ij}$. Matrix G defines the coupling topology, direction, and the coupling strength of the network. N denotes network size. $\delta_i(t)$ and $\phi_i(t)$ are external noise intensity functions, and $\omega(t)$ and $\nu(t)$ are two independent one-dimensional Brownian motions satisfying the mathematical expectations $\mathbb{E}\{d\omega(t)\} = 0$, $\mathbb{E}\{d\nu(t)\} = 0$, $\mathbb{E}\{d\omega(t)^2\} = 1$ and $\mathbb{E}\{d\nu(t)^2\} = 1$. $x_i(t) = \varphi(t) \in \mathcal{C}([-\bar{\tau}, 0], \mathbb{R}^n), \bar{\tau} = \max\{\tau_1, \tau_2\}$ denotes the initial state, which is the set of real-valued continuous functions on $[-\bar{\tau}, 0]$ for some $\tau_1 > 0$ and $\tau_2 > 0$.

6.2. Parameter identification problem

For estimating the unknown parameters of Eq. (29), we assume the structure of system (29) is known. Considering the identified system described by

$$
\begin{aligned}
d\breve{m}_i(t) &= [-\breve{A}\breve{m}_i(t) + \breve{W}f(\breve{p}_i(t-\breve{\tau}_1)) + \breve{L} + \sum_{j=1}^{N} \breve{G}_{ij}\breve{\Gamma}\breve{m}_i(t)]dt + \delta_i(t)d\omega(t), \\
d\breve{p}_i(t) &= [-\breve{C}\breve{p}_i(t) + D\breve{m}_i(t-\breve{\tau}_2) + \sum_{j=1}^{N} \breve{G}_{ij}\breve{\Gamma}\breve{p}_i(t)]dt + \phi_i(t)d\nu(t), \\
i &= 1, 2, \cdots, N.
\end{aligned} \tag{30}
$$

In this paper, the time-delay τ_1, τ_2 and topology configuration matrix G in Eq. (29) are all treated as unknown parameters to be identified. The problem of parameter identification can be converted into the following optimization problem:

$$
J = \sum_{k=1}^{M}\sum_{i=1}^{N} \|m_{ik} - \breve{m}_{ik}\|^2 + \sum_{k=1}^{M}\sum_{i=1}^{N} \|p_{ik} - \breve{p}_{ik}\|^2, \tag{31}
$$

where $k = 1, 2, \cdots, M$ is the sampling time point and M denotes the length of data used for parameter identification. m_{ik}, $\breve{m}_{ik}$, p_{ik} and $\breve{p}_{ik}$ denote the state vector of the original and the identified system at time k, respectively. The parameter identification of system (29) can be achieved by searching suitable γ^*, τ_1^*, τ_2^* and G^* such that the objective function (31) is minimized, i.e.

$$
(\gamma^*, \tau_1^*, \tau_2^*, G^*) = \arg \min_{(\gamma,\tau_1,\tau_2,G)\in\Omega} (J), \tag{32}
$$

where γ is a set of unknown parameters, Ω is searching space admitted for parameters and time delays. It is should be mentioned that it is difficult to identify parameters using traditional optimization methods. One reason is that the dynamic behavior of system (29) is unstable and the other is that there exist multiple variables and local optima in the landscape of J.

Remark 1. *Recently, filtering problem for GRNs has been investigated in some well-studied works [12, 17]. In [17], the authors use an adaptive filtering approach to identify the unknown parameters in GRN. Compared with this work, the time-delays τ_1, τ_2 and coupling matrix of GRNs are also assumed to be unknown and needed to be identified. A novel PSO has been developed in this paper and employed to identify the unknown parameters efficiently in this paper. The simulation results can be seen in next subsection.*

6.3. Example

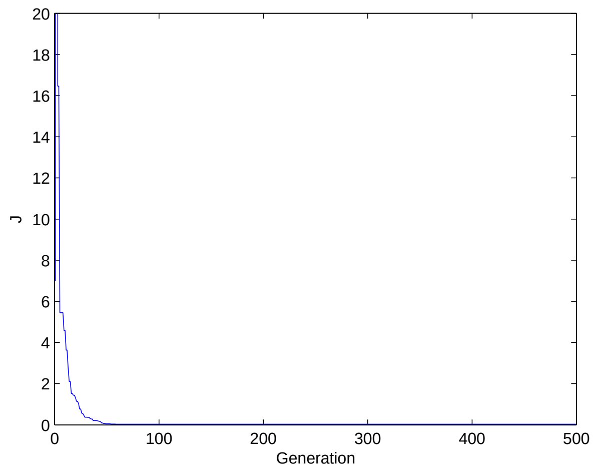


Figure 12: Convergence trajectory of objective function J.

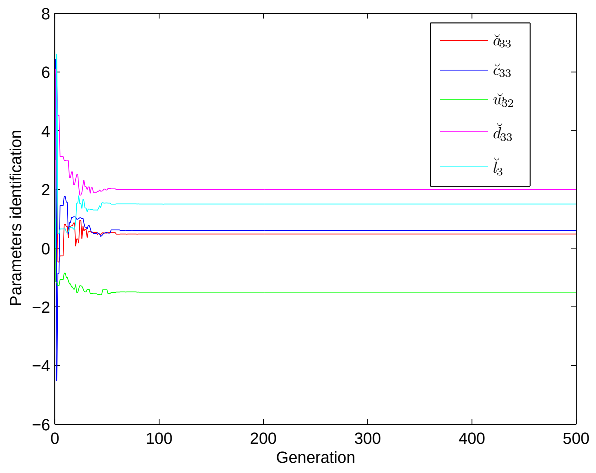


Figure 13: Convergence of $\breve{a}_{33}, \breve{c}_{33}, \breve{w}_{32}, \breve{d}_{33}, \breve{l}_3$.

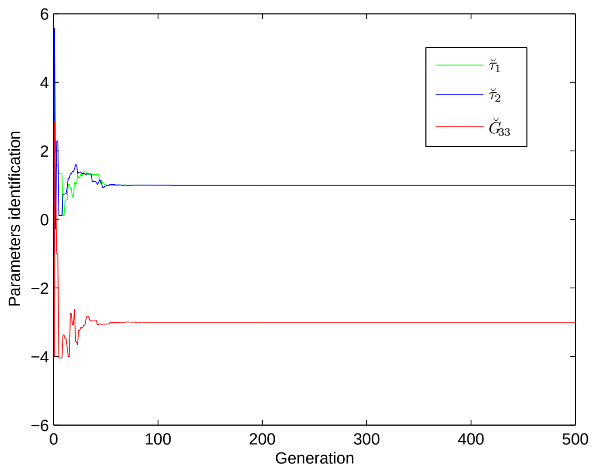


Figure 14: Convergence of $\breve{\tau}_1, \breve{\tau}_2, \breve{G}_{33}$.

In [3], the dynamics of the repressilator has been theoretically investigated and experimentally verified. Three repressor-protein concentrations p_i, and their mRNA concentrations m_i (where i is *lacl*, *tetR* or *cl*) were used as dynamical variables. The repressilator is a cyclic negative-feedback loop composing of three genes and their corresponding promoters. The kinetics of the GRN can be described as follows [8, 9, 17]:

$$\begin{aligned}\frac{dm_i(t)}{dt} &= -m_i(t) + \frac{\alpha}{1+p_j^n(t)} + L_i,\\ \frac{dp_i(t)}{dt} &= -c_i p(t) + d_i m_i(t).\end{aligned} \tag{33}$$

where $i = lacl, tetR, cl$; $j = cl, lacl, tetR$.

Considering the coupling, the stochastic disturbances and transcriptional time delays with GRNs, we introduce the GRNs model as follows:

$$\begin{aligned}&dm_i(t) = [-Am_i(t) + Wf(p_i(t-\tau_1)) + L + \sum_{j=1}^{N} G_{ij}\Gamma m_i(t)]dt + \delta_i(t)d\omega(t),\\ &dp_i(t) = [-Cp_i(t) + Dm_i(t-\tau_2) + \sum_{j=1}^{N} G_{ij}\Gamma p_i(t)]dt + \phi_i(t)d\nu(t),\\ &i = 1, 2, \cdots, N.\end{aligned} \tag{34}$$

The parameters are chosen as follows:

$$A = \begin{pmatrix} 0.4 & 0 & 0 \\ 0 & 0.36 & 0 \\ 0 & 0 & 0.48 \end{pmatrix}, C = \begin{pmatrix} 0.2 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.6 \end{pmatrix}, W = \begin{pmatrix} 0 & 0 & -1.5 \\ -1.5 & 0 & 0 \\ 0 & -1.5 & 0 \end{pmatrix},$$

$$D = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}, G = \begin{pmatrix} -2 & 0 & 2 \\ 0 & -1 & 1 \\ 2 & 1 & -3 \end{pmatrix}, \Gamma = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

Other parameters are taken as $L = (1.5, 1.5, 1.5)^T, \delta_i(t) = \sigma_i(t) = 0.05e^{-0.05t}, f(z) = \frac{z^2}{z^2+1}, \tau_1 = \tau_2 = 1$.

We assume the $a_{33}, c_{33}, w_{32}, d_{33}, l_3, \tau_1, \tau_2, G_{33}$ are unknown, which are needed to identified by SPSO. The population size is set to be 10 and the generation is 500. From Fig.12, a typical evolving process of the objective function J is illustrated and convergence processes of parameters $\breve{a}_{33}, \breve{c}_{33}, \breve{w}_{32}, \breve{d}_{33}, \breve{l}_3, \breve{\tau}_1, \breve{\tau}_2, \breve{G}_{33}$ are shown in Fig. 13 and Fig. 14. Fig. 12 shows that the value of J decreases quickly to zero, which means that SPSO can converge to the global optimum immediately. Furthermore, it can be seen from Fig. 13 and Fig. 14 that the eight unknown parameters can converge to the true values rapidly, which demonstrates the great efficiency of SPSO presented in this paper.

7. Conclusion

The parameter identification problem has been investigated for uncertain GRNs with time-delays and stochastic disturbances by a switching particle swarm optimization algorithm. Depending on a homogeneous Markov chain, the velocity updating equation switches from one mode to another. The current state is determined by evolutionary factor, which is used to predict the next state and determine the acceleration coefficients. Meanwhile, LCPMLA is developed to improve the global search and local search abilities in the evolution process. A learning rate and a penalized rate are designed in this technique aiming at automatically selecting efficient search strategy. The search performance including convergence and accuracy is improved substantially when testing eight benchmark functions. In the end, we have employed the SPSO to identify the unknown parameters of GRNs, which include the unknown coupling matrix and time-delays.

References

[1] J. M. Raser, E.K.O'Shea, Noise in gene expression: Origins, consequences, and control. Science 2005, 309:2010-2013.

[2] T. Chen, H. He, and G. Church, Modeling gene expression with differential equations, in Proc. Pacific Symp. Biocomput., 1999, vol. 4, pp. 29-40.

[3] M. B. Elowitz, S. Leibler: A synthetic oscillatory network of transcriptional regulators. Nature 2000, 403:335-338.

[4] J. Paulsson, Summing up the noise in gene networks, Nature, vol. 427, pp. 415-418, 2004.

[5] P. Smolen, D. Baxter, and J. Byrne, Modelling circadian oscillations with interlocking positive and negative feedback loops, J. Neurosci., vol. 21, pp. 6644-6656, 2001.

[6] P. Smolen, D. Baxter, and J. Byrne, Mathematical modeling of gene networks, Neuron, vol. 26, pp. 567-580, 2000.

[7] AL. Barabási, ZN. Oltvai: Network biology: Understanding the cell's functional organization. Nature Reviews Genetics 2004, 5:101-114.

[8] L. Chen and K. Aihara, Stability of genetic regulatory networks with time delay, IEEE Trans. Circuits Syst. I, vol. 49, no. 5, pp. 602-608, May 2002.

[9] C. Li, L. Chen, and K. Aihara, Stability of genetic networks with sum regulatory logic: Lur'e system and LMI approach, IEEE Trans. Circuits Syst. I, vol. 53, no. 11, pp. 2451-2458, Nov. 2006.

[10] C. Li, L. Chen, and K. Aihara, Stochastic synchronization of genetic oscillator networks, BMC Syst. Biol., vol. 1, no. 6, pp. 1-11, 2007, 10.1186/1752-0509-1-6.

[11] Z. Wang, H. Gao, J. Cao, and X. Liu, On delayed genetic regulatory networks with polytopic uncertainties: robust stability analysis, IEEE Trans. NanoBioscience, vol. 7, no. 2, pp. 154-163, Jun. 2008.

[12] Z. Wang, J. Lam, G. Wei, K. Fraser and X. Liu, Filtering for nonlinear geneti regulatory networks with stochastic disturbances, IEEE Trans. on Automatic Control, Vol. 53, No. 10, Nov. 2008, pp. 2448-2457.

[13] Z. Wang, F. Yang, D. W. C. Ho, S. Swift, A. Tucker and X. Liu, Stochastic dynamic modeling of short gene expression time series data, IEEE Tran. on NanoBioscience, Vol. 7, No. 1, Mar. 2008, pp. 44-55.

[14] Z. Wang, X. Liu, Y. Liu, J. Liang and V. Vinciotti, An extended Kalman filtering approach to modelling nonlinear dynamic gene regulatory networks via short gene expression time series, IEEE/ACM Tran. on Computational Biology and Bioinformatics, in press (Digital Object Identifier: 10.1109/TCBB.2009.5)

[15] F. Ren and J. Cao, Asymptotic and robust stability of genetic regulatory networks with time-varying delays,Neurocomputing, vol. 71, no. 4-6, pp. 834-842, Jan. 2008.

[16] J. Cao and F. Ren, Exponential stability of discrete-time genetic regulatory networks with delays, IEEE Trans. Neural Networks, vol. 19, no. 3, pp. 520-523, 2008.

[17] W. Yu, J. Lü, G. Chen, D. Zhi, Q. Zhou, Estimating Uncertain Delayed Genetic Regulatory Networks: An Adaptive Filtering Approach, IEEE Trans. on Automatic Control, in press.

[18] P. Koduru, S. Das, S. M. Welch, Multi-objective and hybrid PSO using ε-fuzzy dominance, Proc. of GECCO, London, UK, pp. 853-860, 2007.

[19] P. Koduru, Z. Dong; S. Das, S. M. Welch, J.L. Roe, E. Charbit, A Multiobjective Evolutionary-Simplex Hybrid Approach for the Optimization of Differential Equation Models of Gene Networks, IEEE Trans. Evol. Comput., Vol. 12, No. 5, pp.572-590, 2008.

[20] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference On Neural Network, 1995, pp. 1942-1948.

[21] M. Clerc, J. Kennedy, The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space, IEEE Transactions on Evolutionary Computation 6(1). Piscataway, NJ, 2002, pp. 58-73.

[22] Y. Shi, RC. Eberhart. Empirical study of particle swarm optimization. In: Proceedings of the 1999 IEEE congress on evolutionary computation. Piscataway (NJ): IEEE Press; 1999. p. 1945-50.

[23] Y. Shi, RC Eberhart. Parameter selection in particle swarm optimization. In: Proceedings of the 7th international conference on evolutionary programming VII. LNCS, vol. 1447. New York: Springer-Verlag; 1998. p. 591-600.

[24] A. Ratnaweera, SK. Halgamure, HC. Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol. Comput. 2004;8:240-55.

[25] Z. Zhan, J. Zhang, Y. Li, H.S.H. Chung, Adaptive particle swarm optimization, IEEE Trans. System, man and cybernetics-B, in press.

[26] V. Kadirkamanathan, K. Selvarajah, and P. Fleming, Stability Analysis of the Particle Dynamics in Particle Swarm Optimizer, IEEE Trans. on Evol. Comput. Vol. 10, No. 3, 2006, 245-255.

[27] R. Mendes, J. Kennedy, and J. Neves, The fully informed particle swarm: Simpler, maybe better, IEEE Trans. Evol. Comput., vol. 8, no. 3, pp. 204-210, Jun. 2004.

[28] J.J. Liang, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput., vol. 10, no. 3, pp. 281-295, Jun. 2006.

[29] R.A.Krohling and L.dos Santos Coelho, Coevolutionary particle swarm optmization using Gaussian distribution for solving constrned optimization problems, IEEE Trans. Syst., Man, Cybern. B, vol.36, no.6, pp.1407-1416, Dec. 2006.

[30] F. van den Bergh and A. P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. , vol. 8.,no. 3 pp.225-239, June 2004.

[31] R. C. Eberhart and Y. H. SHi, Particle swarm optimization: Developments, applications and resouces, in Proc. IEEE Congr. Evol. Comput. Seoul, Korea, 2001, pp. 81-86.

[32] P. J. Angeline, Using selection to improve particle swarm optimization, in Proc. IEEE Congr. Evol. Comput., Anchorage, AK, 1998, pp. 84-89.

[33] Y. P. Chen, W. C. Peng, and M. C. Jian, Particle swarm optimization with recombination and dynamic linkage discovery, IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 37, no. 6, pp. 1460-1470, Dec. 2007.

[34] P. S. Andrews, An investigation into mutation operators for particle swarm optimization, in Proc. IEEE Congr. Evol. Comput., Vancouver, BC, Canada, 2006, pp. 1044C1051.

[35] Y. Tang, J. Fang, Q. Miao, On the exponential synchronization of stochastic jumping chaotic neural networks with mixed delays and sector-bounded nonlinearties, Neurocomputing, 72 (2009) 1694-1701.

[36] X. Mao and C. Yuan, Stochastic Differential Equations with Markovian Switching (Imperial College Press, 2006).

[37] X. Yao, Y. Liu and G. M. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput., vol.3, no.2, pp.82-102, July,. 1999.

[38] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger and S. Tiwari, Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization, in Proc. IEEE Congr. Evol. Comput.,2005, pp.1-50.

[39] D. H. Wolpert and W. G. Macready, No free lunch theorems for optimization, IEEE Congr. Evol. Comput. vol. 1, no, 1, pp. 67-82, Apr. 1997.