A RESILIENT PARTICLE SWARM OPTIMIZATION ALGORITHM BASED ON CHAOS AND APPLYING IT TO OPTIMIZE THE FERMENTATION PROCESS

LEIFU GAO AND XUWANG LIU

Abstract. When an individual is closed to the optimal particle, its velocity will approximate to zero. This is the main reason why the particle swarm optimization algorithm is prone to trap into local minima, therefore using a strategy in which the velocity is not dependent on the size of distance between the individual and the optimal particle but only dependent on its direction, an adaptive scheme is adopted to adjust the magnitude of the velocity resiliently. At the sametime, by making the best of the ergodicity, stochastic property and regularity of chaos. A resilient particle swarm global optimization algorithm based on chaos is proposed, succeed in applying it to optimize the fermentation process, demonstrate that the new algorithm has the ability to avoid being trapped in local minima, and improves computational precision and convergence ratio.

Key Words. operational research, nonlinear optimization, global optimization, particle swarm, chaos optimization and resilient adjustment.

1. Introduction

The particle swarm optimization (PSO) technique has been developed by Eberhart and Kennedy in 1995 [1] and it is a simple evolutionary algorithm which differs from other evolutionary computation techniques in that it is motivated from the simulation of social behavior[2]. PSO exhibits good performance in finding solutions to static optimization problems. On account of the quick convergence, its realization simply and demanding less conditions for the object-function, for example it doesn’t need grads information, therefore the PSO algorithm develops much speedy and it is succeed in applying during a good many domain. But precocity convergence, local optimization and other disadvantages affect the optimization capability of the PSO algorithm. For solving the disadvantages above the main technique is increasing the swarm’s diversity or inosculating other algorithms at the present time.

The mostly reason of the PSO algorithm’s precocity convergence: at the evening of the algorithm when all the particles congregate at the neighborhood of an extremum point, each particle’s velocity would go to zero. If this is an extremum point, then it will be hardly possible to skip out the restriction area for all the particles, therefore bring on the precocity convergence of the PSO algorithm. The aim of this paper is to propose a resilient particle swarm global optimization algorithm based on chaos (CRPSO). This algorithm makes best use of the chaos and the resilient particle swarm, taking full advantage of the dynamics characteristic.
of chaos, using logistic map to get the beginning Swarm. The logistic map lead to chaotic dynamics. Logistic map is a polynomial map and the chaos sequence has better uniformity and better ergodicity. when calculating the velocity of each particle, we needn’t consider the distance between it with the best particle, but only dependent on its direction, an adaptive scheme is adopted to adjust the magnitude of the velocity resiliently. Therefore When an individual is closed to the optimal particle, its velocity will not approximate to zero, other particles are easy to escape from the restricted area and effectively avoiding precocity convergence. Simultaneity we use the mechanism about precocity judgement and adjust the inertia weight. A resilient particle swarm global optimization algorithm based on chaos is proposed, and then we succeed in applying it to optimize the fermentation process, demonstrate that the new algorithm has the ability to avoid being trapped in local minima, and improves computational precision and convergence ratio.

This paper is organized as follows. In Section 2, The detailed technique and idea of particle swarm optimization are introduced. Section 3 introduces the idea of chaos and the chaos optimization algorithm. A resilient particle swarm global optimization algorithm based on chaos is presented and discussed in Section 4. Then in Section 5, the new CRPSO optimization algorithm is applied in optimizing the fermentation process. Numerical results are presented. Conclusions in Section 6 close the paper.

2. Particle swarm optimization

The Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy [1]. The increasing interest in this method can be observed. It is worth noting that in 2001, about 80 papers on this subject were published, and in the following years, almost a hundred publications each year have been reported. Nowadays, PSO seems to be one of the most commonly used nongradient-based, stochastic search algorithms. In this algorithm, a group of potential solutions, known as particles are flown through a multidimensional search space in order to get an optimum solution. The $i^{th}$ particle in the swarm (a population of particles) is represented as $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$ and its velocity is given by $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$, $i = 1, 2, \cdots, m$. The best particle among all position of each particle is given as $P_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$, the best particle among all particles is represented by a symbol $P_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$.

Hence, a new velocity for each particle is calculated on the basis of its previous value, the particle location at which the best fitness so far has been achieved, and the location at which the best fitness has been achieved for the whole population.

The velocity and position of each particle are updated using following relations:

\begin{align}
(1) & \quad v_{id}(t+1) = wv_{id}(t) + c_1 r_1(p_{id} - x_{id}(t)) + c_2 r_2(p_{gd} - x_{id}(t)) \\
(2) & \quad x_{id}(t+1) = x_{id}(t) + v_{id}(t+1).
\end{align}

where $c_1$ (cognitive constant) and $c_2$ (social constant) are two positive constants, $r_1$ and $r_2$ are functions to generate random number between 0 and 1, and $w$ represents the inertia weight which is used to control between local and global search.

The PSO algorithm has been effectively applied to many research fields like mechanics, computer science, mathematics, or electrical engineering. There are proposals of PSO implementation in multicriteria optimization, integer programming, or approximation methods. Some papers offer modifications of the original algorithm that improve its stability and convergence ability. There are also concepts of hybrid formulations where PSO is combined with other methods like gradient-based
or genetic algorithms. A broad review of PSO papers is given, e.g., by Eberhart and Shi[3-4] or Parsopoulos and Vrahatis[5-6].

3. Chaos optimization

Chaos is now a well established concept and there is an extensive literature on the nature of chaos. Deterministic chaos has proven to be fruitful in the understanding of many complex behavior in the presence of nonlinearity despite the fact that few dimensional systems have actually been found to be low-dimensional deterministic, and thus it influenced thinking in various finance and engineering fields.

For more than three decades, the unusual behavior of chaotic systems has attracted attention of several different scientific communities. Chaotic behaviors have been observed in different fields of sciences as for example engineering, medicine, ecology, biology and economy. Chaos is mathematically defined as randomness generated by simple deterministic systems. In general, chaos has three important dynamic properties[7]:

• the sensitive dependence on initial conditions,
• the semi-stochastic property,
• ergodicity.

Recently, the idea of using chaotic systems instead of random processes has been noticed in several fields. One of these fields is optimization theory. In random-based optimization algorithms, the role of randomness can be played by a chaotic dynamics. Experimental studies assert that the benefits of using chaotic signals instead of random signals are often evident although it is not mathematically proved yet[8]. For example in evolutionary algorithms(EA’s), chaotic sequences increase the value of some measured algorithm-performance indexes with respect to random sequences[9]. Tacking properties of chaos like ergodicity, some new searching algorithms called chaos optimization algorithms(COAs) are presented in literature[10-16]. COA can more easily escape from local minima than other stochastic optimization algorithms[11]. Stochastic optimization algorithm often escape from local minima by accepting some bad solutions according to a certain probability but COA searches on the regularity of chaotic motion to escape from local minima.

We can compare efficiency of different one-dimensional maps as chaotic variable generator in the chaos optimization algorithms. In this regard, we could select 10 different maps and replace the chaos variable generator in one of the existing COAs[16] with them.

This part is organized as follows. In Section 3.1, Problem formulation are introduced. Weighted gradient direction method is presented and discussed in Section 3.2. Then in Section 3.3, the chaos optimization algorithm is introduced.

3.1. Problem formulation. Formally, a nonlinear programming (NLP) problem with inequality constraints can be stated as:

\[
\begin{align*}
\min & \quad f(x) \\
\text{s.t.} & \quad g_i(x) \leq 0, i = 1, 2, \cdots, m, \\
& \quad a_i \leq x_i \leq b_i, i = 1, 2, \cdots, n,
\end{align*}
\]

where \( x = (x_1, x_2, \cdots, x_n)^T \in \mathbb{R}^n \), \( f(x) \) is the objective function and \( g_i(x) \)'s are the inequality constraints defined on \( \mathbb{R}^n \). Also, we assume that functions \( f(x) \) and
The nonlinear programming stated in (3) is converted to another nonlinear programming with only bound constraints

$$\min \ P(x, \sigma) = f(x) + \sigma \sum_{i=1}^{m} \max(0, g_i(x))$$

where \( f(x) \) is the objective function of the original constrained problem and the coefficient \( \sigma \) is a penalty factor. In [17], it was proved that the minima of the non-differentiable exact penalty function (4) converge to the minima of the original constraint problem (3), if penalty parameter \( \sigma \) is chosen sufficiently large.

### 3.2. Weighted gradient direction.

In this section, we introduce weighted gradient direction proposed by Jiang and Wang [18-19]. By using this method, chaos optimization algorithm is improved [16]. Let

$$Q = \{ x \in \mathbb{R}^n | g_i(x) \leq 0, i = 1, 2, \cdots, m \}$$

For an individual \( x \), if \( x \in Q \) moving \( x \) along the negative gradient direction of the objective function \( -\nabla f(x) \), the objective function may be improved. If \( x \notin Q \) it denotes that \( x \) is out of the feasible domain. Assume that

$$I(x) = \{ i | g_i(x) > 0, i = 1, 2, \cdots, m \}$$

For \( i \in I \), by moving \( x \) along the negative gradient direction \( -\nabla g_i(x) \), \( g_i(x) \) can be decreased and may satisfy \( g_i(x) \leq 0 \). Based on these facts, weighted gradient direction is defined as follows:

$$d(x) = -\nabla f(x) - \sum_{i=1}^{n} w_i \nabla g_i(x)$$

\( w_i \) is the weight of gradient direction and defined as follows:

$$w_i = \begin{cases} 0 & g_i(x) \leq 0 \\ \delta_i & g_i(x) > 0 \end{cases}$$

where

$$\delta_i = \frac{1}{1 - \frac{G_i(x)}{G_{\max}(x)+\varepsilon}}$$

(10)$$G_i(x) = \frac{g_i(x)}{||\nabla g_i(x)||}$$

and

(11)$$G_{\max}(x) = \max\{ 0, G_i(x), i = 1, 2, \cdots, m \}$$

\( \varepsilon \) is a very small positive number. Then \( x^{(k+1)} \) is generated from \( x^{(k)} \) by mutation along the weighted gradient direction \( d(x) \) and is described as

$$x^{(k+1)} = x^{(k)} + \beta d(x^k)$$

where \( \beta \) is a positive step-size parameter.

From (7)-(10), it is obvious that for \( g_i(x) > 0 \), increasing of \( g_i(x) \), causes increasing of the weight of gradient direction \( d(x) \) is an effective search direction because if \( x \in Q \), then \( d(x) = -\nabla f(x) \) and it moves along the direction of \( -\nabla f(x) \) consequently the objective function may be improved. Also, if \( x \notin Q \), the bigger the \( g_i(x) \), the further apart \( x \) is from the feasible domain \( Q \), the high weight \( w_i \) can be obtained in order to move into feasible domain. Therefore \( x \) converge to the local optimal solution by weighted gradient direction.
3.3. Chaos optimization algorithm. In previous section, weighted gradient direction is proposed as a local optimization technique for constrained nonlinear problems. By means of ergodicity, regularity and semi-stochastic properties of chaos, the optimal solution migrates in a chaotic way among the local minima and finally converges to the global optimal solution with a high probability.

In the most of COA methods, chaos variables generated by logistic map but in this paper, we do not have any prejudice in favor of using this map. Now, we assume chaos variable can be generated by an arbitrary one-dimensional map. Also, we suppose chaos variables can be change in the range \((0, 1)\). Otherwise we scale chaos variables to this range.

Fig. 1 shows flowchart of the weighted gradient direction based chaos optimization algorithm. Now, we explain components of this flowchart.

3.3.1. Chaos search by using the first carrier wave. In this part we will give the chaos search steps.

Step 1: Initialize the number of the first chaos search \(S_1\), the number of the second chaos search \(S_2\), penalty parameter \(\sigma\), initial value of chaos variables \(0 < r_i(k) < 1, (n = 1, 2, \ldots, n)\) which have small differences, adjusting parameter for small ergodic ranges around solution \(\alpha_i > 0\) and adjusting parameter for \(\alpha t > 1\).

Step 2: Set \(k = 0\) and \(P^* = +\infty\).

Step 3: Map chaos variables \(0 < r_i(k) < 1, (n = 1, 2, \ldots, n)\) into the range of optimization variables by the following equation:

\[
x_i(k) = a_i + r_i(k)(b_i - a_i)
\]

Step 4: If \(P(x(k), \sigma) \leq P^*\) then \(x^* = x_i(k)\) and \(P^* = P(x(k), \sigma)\).

Step 5: Generate next values of chaos variables by a chaotic map function (M):

\[
r_i(k + 1) = M(r_i(k)).
\]

Step 6: \(k \leftarrow k + 1\)

Step 7: If \(k < S_1\), go to step 3, else stop the first chaos search process.

3.3.2. Search along the weighted gradient method. In this part we will give the chaos search steps.

Step 1: Initialize step length \(\beta > 0\) and adjust step length parameters \(t_1 > 1\) and \(0 < t_2 < 1\).

Step 2: Calculate \(d(x^*)\) from (16).

Step 3: \(x = x^* + \beta d(x^*)\).

Step 4: If \(P(x, \sigma) \leq P^*\) then \(x^* = x\) and \(P^* = P(x, \sigma)\) and \(\beta \leftarrow t_2 \beta\) else \(\beta \leftarrow t_1 \beta\).

Step 5: Until \(P^*\) improves go to Step 2.

3.3.3. Chaos search by using the second carrier wave. In this part we will give the chaos search steps.

Step 1: \(k' = 0\)

Step 2: \(x_i(k) = x^* + \alpha_i(r_i(k) - 0.5)\).

Step 3: If \(P(x(k), \sigma) \leq P^*\) then \(x^* = x_i(k)\) and \(P^* = P(x(k), \sigma)\) and stop.

Step 4: Generate next values of chaos variables by a chaotic map function (M):

\[
r_i(k + 1) = M(r_i(k)).
\]

Step 5: \(k \leftarrow k + 1\) and \(k' \leftarrow k' + 1\).

Step 6: If \(k < S_2\) go to step 3.

Step 7: \(\alpha_i \leftarrow t_3 \alpha_i\) and stop the second chaos search process.

\(\alpha\) is a very important parameter and adjusts small ergodic ranges around \(x^*\). It is difficult and heuristic to determine the appropriate value of \(\alpha[16]\). Initial value
of this parameter is usually set to $0.01(b_i - a_i)$. The optimization program is stopped when $\alpha$ becomes more than a specific constant (for example $0.1(b_i - a_i)$).

**Figure 1.** Flowchart of the weighted gradient direction based chaos optimization algorithm

4. The resilient particle swarm global optimization algorithm based on chaos

Suppose the problem to solve as follows:

$$
\text{min } f(x_1, x_2, \cdots, x_n) \text{ s.t. } a_i \leq x_i \leq b_i.
$$

we use logistic map to generate the chaos sequence, the model is:

$$
z_{n+1} = 1 - 2 \times z_n^2, n = 1, 2, \cdots, -1 < z_n < 1.
$$
$w$ is important for the algorithm’s convergence in the PSO algorithm. It makes the particle keep movement inertia, having the trend to extend the search space. Some people proposed adjusting the inertia weight linearity, but we can easily find that from the search character: minishing $w$ linearity would not satisfy the search velocity quickly at the beginning and slowly in the evening. Furthermore the search course of PSO algorithm is nonlinear and quite complicated; so that adjusting the inertia weight linearity would not reflect the actual search course of PSO optimization algorithm. In virtue of the degree of precocity convergence and using individual fitness value to adjust the inertia weight, we combined the chaos with the resilient particle swarm algorithm, proposed a resilient particle swarm global optimization algorithm based on chaos.

4.1. The evaluation of precocity convergence. During the algorithm the global optimization value is always not worse than all the particle’s individual fitness value. Let the sizes of the particle swarm $N$, if $f_{avg}$ is the average value of all the particle’s individual fitness value.

\[
f_{avg} = \frac{1}{N} \sum_{i=1}^{N} f_i.
\]

where $f_i$ is the particle’s fitness value of current iterative time. The best particle’s fitness value among all particles is represented $f_{g}$, then we can get the average value $f'_{avg}$. Define $\Delta = |f_{avg} - f'_{avg}|$, $\Delta$ can estimate the degree of precocity convergence, and it trend to precocity as $\Delta$ is much less.

4.2. Adaptive adjusting strategy. Let $f_i$ be the particle’s individual fitness value, the specific adjusting strategy of the inertia weight as follows:

\[
w = \begin{cases} 
  w - (w - w_{\text{min}}) \cdot \frac{f_i - f_{avg}}{f_g - f_{avg}} & \text{if } f_i \text{ is better than } f_{avg}' \\
  1.5 - \frac{1}{\exp(-k_2 \Delta)} & \text{if } f_i \text{ is worse than } f_{avg}' \\
  w_{\text{min}} + (w_{\text{max}} - w_{\text{min}}) \cdot \frac{1 + \cos((\text{iter}-1)\pi/(\text{max step}-1))}{2} & \text{others}
\end{cases}
\]

where $w_{\text{min}}$ is the least value of $w$, $w_{\text{max}}$ is the maximum when beginning to search. $\text{iter}$ is the iterative steps, $\text{max step}$ is the maximum iterative step. $k_1$ is mainly used to control the upper limit of $w$: the upper limit of $w$ is bigger as $k_1$ is bigger. The selection of $k_1$ should satisfy that the particle has the inertia weight which is bigger than 1. $k_2$ is mainly used to control the modulation capability of upper formula, if $k_2$ is excessive big, $w$ would become to be very small. This can quicken the convergence but it lead to that the searching ability of the algorithm is insufficiency. If $k_2$ is excessive small, the modulation capability of upper formula is not obvious, especially it would not escape from the local minimum at the evening of the algorithm.

4.3. Adjusting the velocity resilient. All the particles of the particle swarm get together at the neighborhood of an extremum point at the evening of algorithm $p_{id}$, $p_{gd}$, $x_{id}$ are almost the same, therefore the velocity $v_{id}$ of each particle would trend to zero. If this extremum point is a local minimum, it is hard to escape from the local minimum for all the particles until all particles concentrate at this extremum point. As a result that will lead to algorithmic’s precocity convergence.

For the sake of improving the searching ability of the algorithm, we need avoid that the velocity of each particle is excessive small. In this paper the direction of the particle swarm is the same as the original algorithm, all pointing to two extremums, but the velocity value is different from the original PSO algorithm.
The velocity is not dependent on the size of distance between the individual and the optimal particle but only dependent on its direction, an adaptive scheme is adopted to adjust the magnitude of the velocity resiliently.

\[ v_{id}(t + 1) = w \ast v_{id}(t) + c_1 \ast r_1 \ast \Delta v_{id}^1(t) + c_2 \ast r_2 \ast \Delta v_{id}^2(t). \]

where

\[
\begin{align*}
\Delta v_{id}^1(t) &= \Delta v_{id}^1(t) \ast \text{sgn}(p_{id}(t) - x_{id}(t)) \\
\Delta v_{id}^2(t) &= \Delta v_{id}^2(t) \ast \text{sgn}(p_{gd}(t) - x_{id}(t))
\end{align*}
\]

\text{sgn}() is the sign function, \( \Delta v_{id}^1(t) \) and \( \Delta v_{id}^2(t) \) is modified resiliently by an adaptive scheme. If the two directions are consistent, it show that the particle is approaching to the extremum from one side, so that we can increase \( \Delta v_{id}^1(t) \) (or \( \Delta v_{id}^2(t) \)) to quicken the convergence of the algorithm. If the two directions are not consistent, it show that the particle is wandering round the extremum, so that we can decrease \( \Delta v_{id}^1(t) \) (or \( \Delta v_{id}^2(t) \)) to avoid its wandering too long round the extremum. If one is the same as the individual extremum (global extremum), don’t change \( \Delta v_{id}^1(t) \) (or \( \Delta v_{id}^2(t) \)), thereupon modifying \( \Delta v_{id}^1(t) \) and \( \Delta v_{id}^2(t) \) as the following formula:

\[ \Delta v_{id}^1(t) = \begin{cases} 
\theta_+ \ast \Delta v_{id}^1(t - 1) & \text{if } |p_{id}(t) - x_{id}(t)| \ast |p_{id}(t - 1) - x_{id}(t - 1)| > 0 \\
\theta_- \ast \Delta v_{id}^1(t - 1) & \text{if } |p_{id}(t) - x_{id}(t)| \ast |p_{id}(t - 1) - x_{id}(t - 1)| < 0 \\
\Delta v_{id}^1(t - 1) & \text{otherwise}
\end{cases} \]

\[ \Delta v_{id}^2(t) = \begin{cases} 
\theta_+ \ast \Delta v_{id}^2(t - 1) & \text{if } |p_{gd}(t) - x_{gd}(t)| \ast |p_{gd}(t - 1) - x_{gd}(t - 1)| > 0 \\
\theta_- \ast \Delta v_{id}^2(t - 1) & \text{if } |p_{gd}(t) - x_{gd}(t)| \ast |p_{gd}(t - 1) - x_{gd}(t - 1)| < 0 \\
\Delta v_{id}^2(t - 1) & \text{otherwise}
\end{cases} \]

where \( \theta_+ \) and \( \theta_- \) are factors increase by degrees and descending, they satisfy \( 0 < \theta_- < 1 < \theta_+ \), at the same time \( \Delta v_{id}^1(t) \) and \( \Delta v_{id}^2(t) \) must be restricted between a reasonable area, \( \Delta v_{id}^1(t), \Delta v_{id}^2(t) \in [\Delta_{\min}, \Delta_{\max}] \).

After we adopt the upside strategy, even though some particles are closed to the best particle, its velocity would not too small, when all particles are attracted at the neighborhood of the local extremum, some particles are able to skip out the restricted area and it is more possible to find the better location, consequently improving the possibility to get the global extremum.

4.4. The global optimization algorithm flow of resilient particle swarm based on chaos. The global optimization algorithm of the resilient particle swarm based on chaos(CRPSO) is based on the PSO optimization algorithm. It adjusts the particle’s velocity according to the flying directions of the former and the later. The particular algorithm flow is as follows:

Step 1: Initialize the correlative parameters, the maximal iterative steps, the error limit of fitness value, the inertia value, the learning factor and so on.

Step 2: Initialize the location and velocity of the particle swarm using chaos maps.

(1) Generate a random numerical value vector of \( \mathbb{R}^n \), \( z_1 = (z_{11}, z_{12}, \ldots, z_{1n}) \) satisfying that \( z_{ij} \in [-1, 1] \), \( n \) is the variable numbers of the objective function. According the iterative map we get vectors \( z_1, z_2, \ldots, z_N \).

(2) Make the subsidiary numerical value be consistent with corresponding value space.

(3) Calculate the fitness value of each particle and then select the best \( M \)-number of the beginning colony, get \( M \)-number random velocity as the beginning velocity.
5. Applying the CRPSO to optimize the biology fermentation process

5.1. The description about the optimization problem. We would optimize the adding ratio of the raw material (microzyme) in the course of producing alcohol batches. Our object is to get the curve of raw material which is the time’s function according to the logistic map \( a \). According to the logistic map \( a \), the adding ratio of the raw material (microzyme) in the course of producing alcohol, the curve relation can maximize the output of the alcohol. So, we suppose that the fermentation time is 54 hours during each production cycle, \( t_f = 54 \). So the mathematic model is described as follows:

The objective function:

\[
\max_u PI = x_3(t_f) \cdot x_4(t_f).
\]

where

\[
\begin{align*}
\frac{dx_1}{dt} &= g_1 x_1 - u(t) \frac{z_1}{x_1} \\
\frac{dx_2}{dt} &= -10 g_1 x_1 + u(t) \frac{150 - x_2}{x_4} \\
\frac{dx_3}{dt} &= g_2 x_2 - u(t) \frac{x_2}{x_4} \\
\frac{dx_4}{dt} &= u(t)
\end{align*}
\]

In the upper formula

\[
\begin{align*}
g_1 &= \frac{0.408}{1 + \frac{x_2}{0.22} + x_2} \\
g_2 &= \frac{1}{1 + \frac{x_2}{0.44}} \frac{x_2}{0.44 + x_2}
\end{align*}
\]

where \( 0 \leq x_4 \leq 200 \) and \( 0 \leq u(t) \leq 12 \).
The beginning state: \( [x_1(0), x_2(0), x_3(0), x_4(0)] = [1, 150, 0, 10]^T \).

In the upper formula, \( x_1 \) is the weight of the thalli. \( x_2 \) and \( x_3 \) are the consistency of the raw material and the production. \( x_4 \) is the volume of the product. \( 0 \leq x_4 \leq 200 \) must be satisfied during \([0, t_f]\). Our object is to maximize the output of the alcohol in the fermentation jar.

5.2. The particular optimization result. Firstly we translate the restrictive optimization problem into abandoned optimization problem, according to the punishing function method, its objective function is translated into that:

\[
\min_u PI = -x_3(t_f) \cdot x_4(t_f) + \alpha \cdot [\min(0, 200 - x_4(t_f))]^2
\]

Secondly enactment the relevant parameters: \( \text{pop} = 100; \alpha = 10^6; \theta_+ = 1.1 \) and \( \theta_- = 0.5 \). \( \Delta(0) \in [0.01 \Delta u, 0.1 \Delta u] \); \( c_1 = c_2 = 2.0; \Delta_{\text{max}} = 0.5 \Delta u, \Delta_{\text{min}} = 0.001 \Delta u \). The control periods is 3 hours, we optimize the fermentation process making use of the CRPSO which this paper proposed, through 300 steps’ iteration, the performance object value \( \min PI = -19932 \) and \( V(t_f) = x_4 = 200 \).

Optimal feed rate of a fed-batch bioreactor for the production of ethanol which was optimized by CRPSO can been seen in table 1. The compare curve of the optimal performance between CRPSO and CPSO can been seen in figure 2.

<table>
<thead>
<tr>
<th>fermentation time(h)</th>
<th>0.0000</th>
<th>3.0000</th>
<th>6.0000</th>
<th>9.0000</th>
<th>12.000</th>
<th>15.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>adding ratio(l/h)</td>
<td>0.5550</td>
<td>0.3113</td>
<td>0.6758</td>
<td>0.1023</td>
<td>1.2536</td>
<td>3.0121</td>
</tr>
<tr>
<td>fermentation time(h)</td>
<td>18.000</td>
<td>21.000</td>
<td>24.000</td>
<td>27.000</td>
<td>30.000</td>
<td>33.000</td>
</tr>
<tr>
<td>adding ratio(l/h)</td>
<td>0.1402</td>
<td>2.2032</td>
<td>4.3945</td>
<td>0.7300</td>
<td>5.2510</td>
<td>9.5865</td>
</tr>
<tr>
<td>fermentation time(h)</td>
<td>36.000</td>
<td>39.000</td>
<td>42.000</td>
<td>45.000</td>
<td>48.000</td>
<td>51.000</td>
</tr>
<tr>
<td>adding ratio(l/h)</td>
<td>5.5362</td>
<td>10.298</td>
<td>11.152</td>
<td>4.1588</td>
<td>3.9833</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

We can find that from the figure: the CRPSO algorithm can achieve the global minimum after 120 iterations but the CPSO algorithm need 220 iterations, simultaneity the CPSO algorithm would get into the local minimum easily. The new CRPSO algorithm has better computational precision and has better convergence ratio. The numerical simulations results demonstrated that the new algorithm would save lots of resource and increase economy benefit when it is taken into practice.

6. Conclusions

Based on the analysis about the precocity convergence of the particle swarm optimization, making use of changing the particle’s velocity resiliently, this paper combined the chaos with the resilient particle swarm algorithm, simultaneity we used the mechanism about precocity judgement to adjust the inertia weight, proposed a resilient particle swarm global optimization algorithm based on chaos. The new algorithm enhanced the multiplicity of the particle swarm from lots of sides, effectively avoided precocity convergence. and improved computational precision and convergence ratio. In succession we succeed in applying the new algorithm to optimize the fermentation process, the numerical simulations results demonstrated that the new algorithm has the ability to avoid being trapped in local minima and effective. But excessive parameters bring inconvenience to the algorithm, modifying the parameters adaptively is our research content later.
Acknowledgments

The authors thank the anonymous referees for their very careful reviews of the paper.

References


Institute of Mathematics and Systems Science, University of Liaoning Technical, Liaoning, Fuxin 123000, CHINA

E-mail: gaoleifu@yahoo.com.cn and liuxuwang007@163.com