Chaotic Multi-Objective Particle Swarm Optimization Algorithm Incorporating Clone Immunity

Ying Sun 1, Yuelin Gao 2,* and Xudong Shi 3

1 School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China; sunying@hnu.edu.cn
2 Ningxia Province Key Laboratory of Intelligent Information and Data Processing, North Minzu University, Yinchuan 750021, China
3 School of Mathematics and Statistics, Ningxia University, Yinchuan 750021, China; 2008021@nun.edu.cn
* Correspondence: 1993001@nun.edu.cn; Tel.: +86-951-2066579

Received: 22 November 2018; Accepted: 28 January 2019; Published: 3 February 2019

Abstract: It is generally known that the balance between convergence and diversity is a key issue for solving multi-objective optimization problems. Thus, a chaotic multi-objective particle swarm optimization approach incorporating clone immunity (CICMOPSO) is proposed in this paper. First, points in a non-dominated solution set are mapped to a parallel-cell coordinate system. Then, the status of the particles is evaluated by the Pareto entropy and difference entropy. At the same time, the algorithm parameters are adjusted by feedback information. At the late stage of the algorithm, the local-search ability of the particle swarm still needs to be improved. Logistic mapping and the neighboring immune operator are used to maintain and change the external archive. Experimental test results show that the convergence and diversity of the algorithm are improved.

Keywords: multi-objective optimal problem; particle swarm optimization; clone immunity; chaotic sequence

1. Introduction

Most problems [1–6] in engineering and science are multi-objective optimization problems, and these objectives usually conflict with each other. The focus of academics and engineers is on finding an optimal solution for these problems. For multi-objective optimization problems, however, when a certain objective is improved, it often simultaneously impacts other objectives. Usually, people make a trade-off and compromise to solve this problem. A very effective method to solve such problems is by using an intelligent algorithm. In recent years, intelligent algorithms have been paid increasing attention in many areas [7–10] and have achieved good results. However, when solving multi-objective optimization problems using intelligent algorithms, there are three goals that need to be satisfied: (1) particles must be as close to the Pareto optimal front as possible; (2) there must be a maximal number of particles in the Pareto optimal front; and (3) particles must be distributed as evenly as possible in the Pareto optimal front.

The particle swarm optimization (PSO) algorithm [11,12] is one of the most important and studied paradigms in computational swarm intelligence. It was put forward by Dr. Kennedy and Professor Eberhart in 1995. The simple form, simple parameters, and rapid convergence of PSO have contributed to its rapid development and it becoming a research hotspot over the last 20 years. For multi-objective PSO algorithms, there are six research directions [13]: (1) aggregating approaches, which combine all the objectives of the problem into a single one. In other words, the multi-objective problem is transformed into a single-objective problem. It is not a new idea, since aggregating functions can be derived from the well-known K-T conditions for non-dominated solutions. (2) Lexicographic
ordering, in which the objectives are ranked in order of importance. The optimal solution is obtained by separately minimizing the objective functions, starting with the most important one and proceeding according to the order of importance of the objectives. Lexicographic ordering tends to be useful only when few objective functions are used, and it maybe sensitive to the ordering of the objectives. (3) Subpopulation approaches, which use several subpopulations as single-objective optimizers. In order to balance the objectives, the subpopulations exchange information with each other. Approaches: The information exchange with Sub-Population is not well controlled, and it is difficult to ensure concurrent evolution of each objective. (4) Pareto-based approaches, which use leader selection techniques based on Pareto dominance. This method is the mainstream method, such as multi-objective particle swarm (MOPSO) [14]. It is useful for the multi-objective optimization problems with few objective functions, but when the objective functions increase, the selection pressure increases. (5) Combined approaches, that combine the PSO algorithm with other algorithms. Some examples include the genetic algorithm (GA) [15], the cultural algorithm [16], and the differential evolution (DE) algorithm [17]. (6) Other approaches [18,19] that do not fall into the above five types, such as designing a threshold for the particle updating strategy [20].

The remainder of this paper is organized as follows. In Section 2, we describe the multi-objective optimization and PSO algorithm. Thereafter, in Section 3, we explain a computational method to improve the multi-objective PSO algorithm. Section 4 outlines the strategy and flow of this algorithm. Test problems, performance measures, and results are provided in Section 5, and conclusions are presented in Section 6.

2. Propaedeutics

2.1. Multi-Objective Optimization Problem

In general, a multi-objective optimization problem with $n$ decision variables and $M$ objective functions can be described as [21]:

$$\begin{align*}
\text{min} & \quad y = f(x) = [f_1(x), f_2(x), \cdots, f_M(x)] \\
\text{s.t.} & \quad g_i(x) \leq 0, \quad i = 1, 2, \cdots, p \\
& \quad h_j(x) = 0, \quad j = 1, 2, \cdots, q
\end{align*}$$

(1)

where, $x = (x_1, x_2, \cdots, x_n) \in X$ is a single decision variable, $X$ is the decision variable space, $y = (f_1, f_2, \cdots, f_M) \in Y$ is the objective function vector, $Y$ is the objective function space, $g_i(x)$ is the $i^{th}$ inequality constraint, and $h_j(x)$ is the $j^{th}$ equality constraint.

The following definitions are useful for the conceptual framework of a multi-objective optimization problem:

**Definition 1.** (Pareto dominance) Vector $x' = (x'_1, x'_2, \cdots, x'_n)$ dominates vector $x = (x_1, x_2, \cdots, x_n)$ if and only if the next statement is true:

$$\forall i = (1, 2, \cdots, n) : f_i(x') \leq f_i(x) \quad \text{and} \quad \exists i \in (1, 2, \cdots, n) : f_i(x') < f_i(x).$$

We denote this dominance as $x' \prec x$.

**Definition 2.** (Pareto optimality) A solution $x^* \in D$ is a Pareto optimal solution if there is no other $x \in D$ that satisfies $f(x) \prec f(x^*)$.

**Definition 3.** (Pareto optimal set) The Pareto optimal set is defined as the set of all Pareto optimal solutions.

**Definition 4.** (Pareto optimal front) The Pareto optimal front consists of the values of the objective functions at the solutions in the Pareto optimal set.
2.2. Particle Swarm Optimization Algorithm

Let \( n \) be the dimension of the search space, \( x_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \) be the current position of the \( i \)th particle in the swarm, \( p_{i, \text{best}} = (p_{i1, \text{best}}, p_{i2, \text{best}}, \ldots, p_{in, \text{best}}) \) be the best position of the \( i \)th particle at that time, and \( g_{\text{best}} = (g_{1, \text{best}}, g_{2, \text{best}}, \ldots, g_{n, \text{best}}) \) be the best position that any particle in the entire swarm has visited. The rate of the velocity of the \( i \)th particle is denoted as \( v_{i} = (v_{i1}, v_{i2}, \ldots, v_{in}) \). The position of each particle \( x_{ij} \), and its velocity \( v_{ij} \) updated according to the following:

\[
\begin{align*}
v_{ij}^{t+1} &= w v_{ij}^t + c_1 r_1 (p_{ij, \text{best}}^t - x_{ij}^t) + c_2 r_2 (g_{ij, \text{best}}^t - x_{ij}^t) \quad \text{(2)} \\
x_{ij}^{t+1} &= x_{ij}^t + v_{ij}^{t+1} \quad \text{(3)}
\end{align*}
\]

where \( c_1, c_2 \) are the learning factors, and \( r_1, r_2 \) are random numbers in \([0, 1]\). \( w \) is the inertia weight, which is defined as follows [22]:

\[
w = w_{\text{max}} - t * (w_{\text{max}} - w_{\text{min}}) / T_{\text{max}} \quad \text{(4)}
\]

where, \( w_{\text{max}}, w_{\text{min}} \) are the maximum and minimum of inertia weight \( w \), respectively, \( t \) is the time of the present iteration, and \( T_{\text{max}} \) is the maximum number of iterations.

3. Pareto Entropy and Difference Entropy-Based Improvements of PSO

3.1. Parallel Cell Coordinate System

The parallel cell coordinate system (PCCS) [23] maps the target vector of non-dominated solutions in the external archive to a two-dimensional plane by parallel coordinates, and then rounds these coordinate values. Its mathematical formula is as follows:

\[
L_{k,m} = \left\lceil K f_{k,m} - f_{m, \text{min}} \right\rceil \\
f_{m, \text{max}} = \max_{k} f_{k,m}, \quad f_{m, \text{min}} = \min_{k} f_{k,m}
\]

where \( \lceil x \rceil \) is a top integral function; \( k = 1, 2, \ldots, K \) where \( K \) is the external archive size in the current iteration; \( m = 1, 2, \ldots, M \) where \( M \) is the number of objective functions in the optimization problem; \( f_{m, \text{max}} = \max_{k} f_{k,m} \), and \( f_{m, \text{min}} = \min_{k} f_{k,m} \). When \( f_{m, \text{min}} = f_{m, \text{max}} \), \( L_{k,m} = 1 \).

Consider Table 1 as an example, in which eight particles were included for three objectives. According to the formula (5), we can calculate \( L_{k,m} \) and draw the Figure 1.

<table>
<thead>
<tr>
<th>tab</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>L_{k,1}</th>
<th>L_{k,2}</th>
<th>L_{k,3}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5377</td>
<td>1.8339</td>
<td>−2.2588</td>
<td>4</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.8622</td>
<td>0.3188</td>
<td>−1.3077</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>−0.4336</td>
<td>0.3426</td>
<td>3.5784</td>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>2.7694</td>
<td>−1.3499</td>
<td>3.0349</td>
<td>8</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>0.725</td>
<td>−0.0631</td>
<td>0.7147</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>−0.2050</td>
<td>−0.1241</td>
<td>1.4897</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>1.4090</td>
<td>1.4172</td>
<td>0.6715</td>
<td>6</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>−1.2075</td>
<td>0.7172</td>
<td>1.6302</td>
<td>1</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>
Figure 1. Particle distribution in parallel cell coordinates system.

In this paper, we use the Pareto entropy to estimate the distribution uniformity of the Pareto front, which is calculated with the following formula:

$$\text{Entropy}(t) = - \sum_{k=1}^{K} \sum_{m=1}^{M} \frac{\text{Cell}_{k,m}(t)}{KM} \log \frac{\text{Cell}_{k,m}(t)}{KM}$$  \hspace{1cm} (6)

When approximating the Pareto front map to the PCCS, \(\text{Cell}_{k,m}(t)\) is the number of cell coordinate components in the \(k\)th row and \(m\)th column cell.

According to Pareto entropy, we know the distribution of particles in the current iteration. However, in order to judge the change between this iteration and the last iteration, a difference entropy \(\Delta \text{Entropy}\) is proposed [23]:

$$\Delta \text{Entropy} = \text{Entropy}(t) - \text{Entropy}(t - 1)$$  \hspace{1cm} (7)

Through the difference entropy, information about the external archive can be obtained, and we can dynamically adjust population using this information.

3.2. Difference Entropy Discussion

Previous work has proved that the maximum Pareto entropy is \(\log M\) and the minimum Pareto entropy is \(\log KM\) [23]. Here, we discuss a few special situations for the difference entropy \(\Delta \text{Entropy}\):

(1) In the \(t^{th}\) iteration, the corresponding coordinate components in the PCCS of all objective vectors on the Pareto front are the most nonuniform, and after maintaining the external archive, the corresponding coordinate components of all objective vectors are the most uniform.

**Proof.** Assume that, in the \(t^{th}\) iteration, the distribution of parallel cell coordinates is \(\text{Cell}_{k,m}(t)\) and the external archive size is \(K_1\). In the \((t + 1)^{th}\) iteration, the distribution of parallel cell coordinates is \(\text{Cell}_{k,m}(t + 1)\), and the external archive size is \(K_2\). Next, make all components of the \(n\)th column collect in the \(n\)th column of \(c\)-row, so that \(\text{Cell}_{c,n}(t) = K_1\) and \(\text{Cell}_{c,n}(t + 1) = 0\). From the L’Hospital’s rule, we can know \(\lim_{x \to 0} x \log(x) = 0\), so we stipulate that \(0 \log(0) = 0\). In the \((t + 1)^{th}\) iteration, \(\text{Cell}_{k,m}(t + 1) = 1\), that is to say, it always holds for \(k\)th row and \(m\)th column. We obtain Difference Entropy through the following formula:
\[ \Delta \text{Entropy} = \text{Entropy}(t + 1) - \text{Entropy}(t) \]
\[ = - \sum_{k=1}^{K_2} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t+1)}{K_2^M} \log \frac{\text{Cell}_{km}(t+1)}{K_2^M} - \left( - \sum_{k=1}^{K_1} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t)}{K_1^M} \log \frac{\text{Cell}_{km}(t)}{K_1^M} \right) \]
\[ = \sum_{k=1}^{K_1} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t)}{K_1^M} \log \frac{\text{Cell}_{km}(t)}{K_1^M} - \sum_{k=1}^{K_2} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t+1)}{K_2^M} \log \frac{\text{Cell}_{km}(t+1)}{K_2^M} \]
\[ = \sum_{m=1}^{M} \frac{K_1}{K_1^M} \log \frac{K_1}{K_1^M} - \sum_{m=1}^{M} \frac{K_2}{K_2^M} \log \frac{K_2}{K_2^M} \]
\[ = \log \frac{1}{K_1^M} - \log \frac{1}{K_2^M} \]
\[ = \log K_2 \]

\( \square \)

(2) In the \( t^{th} \) iteration, the corresponding coordinate components in the PCCS of all objective vectors on the Pareto front are the most nonuniform and after maintaining the external archive, the corresponding coordinate components of all objective vectors remain the most nonuniform.

**Proof.** From Case (1), we know \( \text{Cell}_{c_1,n_1}(t) = K_1 \) and \( \text{Cell}_{k \neq c_1,n_1}(t) = 0 \). Furthermore, \( \text{Cell}_{c_2,n_2}(t+1) = K_2 \) and \( \text{Cell}_{k \neq c_2,n_2}(t+1) = 0 \). From Equation (7):

\[ \Delta \text{Entropy} = \text{Entropy}(t + 1) - \text{Entropy}(t) \]
\[ = - \sum_{k=1}^{K_2} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t+1)}{K_2^M} \log \frac{\text{Cell}_{km}(t+1)}{K_2^M} - \left( - \sum_{k=1}^{K_1} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t)}{K_1^M} \log \frac{\text{Cell}_{km}(t)}{K_1^M} \right) \]
\[ = \sum_{k=1}^{K_1} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t)}{K_1^M} \log \frac{\text{Cell}_{km}(t)}{K_1^M} - \sum_{k=1}^{K_2} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t+1)}{K_2^M} \log \frac{\text{Cell}_{km}(t+1)}{K_2^M} \]
\[ = \sum_{m=1}^{M} \frac{K_1}{K_1^M} \log \frac{K_1}{K_1^M} - \sum_{m=1}^{M} \frac{K_2}{K_2^M} \log \frac{K_2}{K_2^M} \]
\[ = \log \frac{1}{K_1^M} - \log \frac{1}{K_2^M} \]
\[ = 0 \]

\( \square \)

(3) In the \( t^{th} \) iteration, the corresponding coordinate components in the PCCS of all objective vectors on the Pareto front are the most uniform and after maintaining the external archive, the corresponding coordinate components of all objective vectors remain the most uniform.

**Proof.** From Cases (1) and (2), we know that \( \text{Cell}_{km}(t) = 1 \) and \( \text{Cell}_{km}(t+1) = 1 \). So:

\[ \Delta \text{Entropy} = \text{Entropy}(t + 1) - \text{Entropy}(t) \]
\[ = - \sum_{k=1}^{K_2} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t+1)}{K_2^M} \log \frac{\text{Cell}_{km}(t+1)}{K_2^M} - \left( - \sum_{k=1}^{K_1} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t)}{K_1^M} \log \frac{\text{Cell}_{km}(t)}{K_1^M} \right) \]
\[ = \sum_{k=1}^{K_1} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t)}{K_1^M} \log \frac{\text{Cell}_{km}(t)}{K_1^M} - \sum_{k=1}^{K_2} \sum_{m=1}^{M} \frac{\text{Cell}_{km}(t+1)}{K_2^M} \log \frac{\text{Cell}_{km}(t+1)}{K_2^M} \]
\[ = \sum_{m=1}^{M} \frac{1}{K_1^M} \log \frac{1}{K_1^M} - \sum_{m=1}^{M} \frac{K_2}{K_2^M} \log \frac{1}{K_2^M} \]
\[ = \log \frac{1}{K_1^M} - \log \frac{1}{K_2^M} \]
\[ = \log \frac{K_2}{K_1} \]

\( \square \)

(4) In the \( t^{th} \) iteration, the corresponding coordinate components in the PCCS of all objective vectors on the Pareto front are the most uniform, but after maintaining the external archive, the corresponding coordinate components of all objective vectors are the most nonuniform.
Proof. From Case (1), we know $Cell_{k,m}(t) = 1$, $Cell_{c_2,n_2}(t+1) = K_2$, and $Cell_{k\neq c_2,n_2}(t+1) = 0$.

$$\triangle \text{Entropy} = \text{Entropy}(t+1) - \text{Entropy}(t)$$

$$= - \frac{K_1}{K_1 M} \sum_{k=1}^{M} \log \frac{Cell_{k,m}(t)}{K_1 M} - \left( - \frac{K_1}{K_1 M} \sum_{k=1}^{M} \frac{Cell_{k,m}(t)}{K_1 M} \log \frac{Cell_{k,m}(t)}{K_1 M} \right)$$

$$= \frac{K_1}{K_1 M} \sum_{k=1}^{M} \log \frac{1}{K_1 M} - \frac{K_2}{K_2 M} \log \frac{K_2}{K_2 M}$$

$$= - \log \frac{1}{K_1 M} - \log \frac{1}{M}$$

$$= \log \frac{1}{K_1}$$

\[ (11) \]

In general, there are only four cases for maintaining the external archive, that is, nonuniform to more nonuniform, nonuniform to relatively uniform, uniform to relatively nonuniform, and uniform to more uniform. It is certain that difference entropy has a maximum and minimum, that is to say, the difference entropy is bounded.

3.3. Sketch of State Inspection

The distribution of particles in an external archive can directly react with entropy, and the status of the external archive can be indirectly speculated by the difference entropy. This reaction may be a convergence condition, diverse condition, or stagnant; then, we can change the parameters of the algorithm through the feedback information of these particles. So, the primary question is how to test the state of particles. We directly use the threshold values from the literature [21]:

$$\delta_c = \frac{2}{H} \log 2$$

$$\delta_s = \frac{2}{MK} \log 2,$$

where $H$ is the number of elements in the external archive, $K$ is the maximum capacity of the external archive, and $M$ is the number of objective functions.

Now, the determining conditions of the environment are given as follow:

- Convergence condition: $|\triangle \text{Entropy}| > \delta_c$ or $|H(t) - H(t-1)| > 0$
- Divergence condition: $\delta_i < |\triangle \text{Entropy}| < \delta_c$ and $H(t) = H(t-1) = K$
- Stagnant condition: $|\triangle \text{Entropy}| < \delta_s$ and $H(t) = H(t-1)$

Obviously, the determining conditions are sensitive to changes in $H$. Whenever $H$ increases or decreases, it is seen as a convergence condition because, when $H$ changes, at least a non-dominated solution is introduced from the external archive. The external archive changes, and the PCCS may also change. This is because an $H$ increase or decrease may lead to the maximum or minimum of one dimension in the objective vector change.

4. Multi-Objective Particle Swarm Optimization Algorithm Based on Clone Immunity

4.1. External Archive Update

In a multi-objective particle swarm optimization algorithm, the maintenance of the external archives plays a key role. First, the calculation method of the individual density is introduced [23]. When the particle is mapped to the PCCS, the individual density of $P_i(i, 1 = 2, \cdots, K, K$ is the external archive size) is obtained according to the following formula:

$$\text{Density}(P_i) = \frac{1}{\sum_{j=1, j\neq i}^{K} \text{PCD}(P_i, P_j)^2}$$

(14)
where \( P_j(j = 1, \ldots, K, j \neq i) \) is parallel cell distance between \( P_i \) and \( P_j \), which can be calculated by the following formula:

\[
PCD(P_i, P_j) = \begin{cases} 
0.5 & \text{if } \forall m, L_{i,m} = L_{j,m} \\
\frac{1}{M} \sum_{m=1}^{M} |L_{i,m} - L_{j,m}| & \text{if } \exists m, L_{i,m} \neq L_{j,m}
\end{cases}
\]  

(15)

Then, the specific update steps in Algorithm 1 [23].

**Algorithm 1**: Improved external archive updating algorithm.

**Input**: (i) External archive \( A \).

(ii) Maximum size of the external archive \( K \).

(iii) New solution \( P \) obtained by the algorithm.

**Output**: Updated external archive \( A_{\text{new}} \).

**Step 1**: Determine \( B = A \cup P \) and calculate the objective vector value of \( B \).

**Step 2**: Find the non-dominated solution set \( B' \) for \( B \).

**Step 3**: If \( |B'| \leq K \), then \( A_{\text{new}} = B' \); if not, turn to Step 4.

**Step 4**: Calculate the individual density of all particles in \( B' \).

**Step 5**: Place particles in ascending order by their individual density, and take the first \( K \) particles to compose \( A_{\text{new}} \).

4.2. Update Strategy of the Global Best Position and Personal Best Position

Particles are selected by the lattice dominance strength [23], the specific steps in Algorithm 2.

**Algorithm 2**: Update strategy of the global best position.

**Input**: (i) External archive \( A \) and the number of objective functions \( M \).

(ii) Status of the external archive.

**Output**: Global best position.

**Step 1**: Calculate the lattice coordinate vector \( L_{k,m} \) of particles in the external archive.

**Step 2**: Calculate the lattice dominant strength and individual density \( \text{Density}(i) \).

**Step 3**: Determine \( a \) and \( b \) by the status of the external archive.

(i) convergence: \( a = M - 1, b = M + 1 \).

(ii) diversification: \( a = M + 1, b = M - 1 \).

(iii) stagnation: \( a = M, b = M \).

**Step 4**: Sort the particles in descending order by the lattice dominant strength and in ascending order by the individual density.

**Step 5**: Take the first \( a \) particles by the individual density and the first \( b \) particles by the lattice dominant strength to store in \( C \).

**Step 6**: Randomly select one particle from \( C \) as the global best position.

For the update strategy of the personal best position, we adopted the following method: [24]:

\[
p_{i,\text{best}}^{t} = \begin{cases} 
p_{i,\text{best}}^{t-1} & \text{if } f(p_{i,\text{best}}^{t}) \prec f(x_{i}^{t}) \\
x_{i}^{t} & \text{if } f(x_{i}^{t}) \prec f(p_{i,\text{best}}^{t}) \\
\text{randselect}(p_{i,\text{best}}^{t-1}, x_{i}^{t}) & \text{otherwise}
\end{cases}
\]  

(16)

4.3. Parameter Selection Strategy

The PSO algorithm has a simple form and can be easily implemented, but always approaches the global and personal best position, and tends to a local best position after several iterations. In order
to overcome the early-maturing phenomenon, the adaptive inertia weight based on the feedback information from status evaluation is used in this paper [25].

\[
w(t) = \begin{cases} 
  w(t-1) & \text{stagnation} \\
  w(t-1) - 2 \times \text{Step}_w(1 + |\Delta \text{Entropy}(t)|) & \text{convergence} \\
  w(t-1) + 2 \times \text{Step}_w(|\Delta \text{Entropy}(t)|) & \text{diversification}
\end{cases}
\]  

(17)

where \( \text{Step}_w \) is the adjusting stepsize of \( w \), and its formula is as follows:

\[
\text{Step}_w = \frac{w_{\text{max}} - w_{\text{min}}}{T_{\text{max}}},
\]

(18)

where \( w_{\text{max}} = 0.9 \), \( w_{\text{min}} = 0.4 \), and \( T_{\text{max}} \) is the maximum number of iterations. The initial value of \( w \) is 0.9.

For \( c_1 \) and \( c_2 \), nonlinear functions of the inertia weight \( w \) are used [23]:

\[
\begin{align*}
  c_1(t) &= 1.167 \times w^2 - 0.1167 \times w + 0.66 \\
  c_2(t) &= 3 - c_1(t)
\end{align*}
\]

(19)

Thus, the learning factors \( c_1 \) and \( c_2 \) also adjust dynamically with the adjustment of the inertia weight.

According to Figure 2, the inertia weight decreased approximately linearly. With the inertia weight change, the learning factor changed.

4.4. Clone Immune Strategy

With each iteration of the algorithm, the diversity of the population decreases. In our paper, the clone, recombination, and mutation operations are used to solve this problem.

(1) Clone: For the external archive \( A \), the population was cloned according to the crowding distance [26]. The larger the crowding distance, the larger the clone population. Then, the specific steps of clone in Algorithm 3:
Algorithm 3: Clone algorithm.

**Input:** (i) External archive $A$ and number of objective functions $M$.
(ii) Maximum size of the clone population $N_C$.

**Output:** Clone population $A'$.

**Step 1:** Calculate the crowding degree of the active population: $padis$.

**Step 2:** Calculate the number of clones of each particle with the following formula:

$$q_i = \left\lfloor NC \times \frac{padis_i}{\sum_{j=1}^{A} padis_j} \right\rfloor.$$  

(20)

**Step 3:** If the total number of clone particles exceeds $N_C$, add the first $N_C$ particles to $A'$ and return $A'$.

Sparse areas get more clone particles by the cloning operation, which can be performed several times.

(2) **Recombination:** Clone population $A'$ is recombined [26], the specific steps in Algorithm 4:

Algorithm 4: Recombination algorithm.

**Input:** (i) Clone population $A'$ and External archive $A$.
(ii) Lower bound of particles: $x_{\text{min}}$, upper bound of particles: $x_{\text{max}}$, the size of the clone population: $N'$, and parameter $\epsilon, r_1$.

**Output:** Recombination population $A''$.

**Step 1:** For each particle $x'(i) \in A'$, randomly select a particle $x(t) \in A$.

**Step 2:** Generate a random number $\text{rand} \in [0, 1]$. If $\text{rand} < r_1$, calculate $\Delta = |x'(i)_j - x(t)_j|$, $y_1 = \min(x'(i)_j, x(t)_j)$, $y_2 = \max(x'(i)_j, x(t)_j)$.

**Step 3:** If $\Delta > \epsilon$,

- **Step 3.1** If $y_1 - x_{\text{min}j} > x_{\text{max}j} - y_2$, then $N_{\text{beta}} = 1 + \frac{2 \times (x_{\text{max}j} - y_2)}{y_2 - y_1}$.
- **Step 3.2** Else $N_{\text{beta}} = 1 + \frac{2 \times (y_1 - x_{\text{min}j})}{y_2 - y_1}$.

**Step 4:** Let $\beta = \frac{1}{N_{\text{beta}}}$ and $\alpha = 2 - \beta e^\phi$.

**Step 5:** Generate a random number $r \in [0, 1]$, if $r < \frac{1}{\alpha}$, then $\alpha = \alpha \times r$, else, $\alpha = 2 - \alpha \times r$.

**Step 6:** Let $e^\phi = \frac{1}{16}$ and $N_{\text{beta}} = \alpha e^\phi$.

**Step 7:** Calculate

$$chld_1 = 0.5 \times [(y_1 + y_2) - N_{\text{beta}} \times (y_2 - y_1)], chld_2 = 0.5 \times [(y_1 + y_2) + N_{\text{beta}} \times (y_2 - y_1)].$$

**Step 8:** Generate a random number $\text{rand} \in [0, 1]$, if $\text{rand} < r_2$, then $x'(i)_j = \min(\max(chld_1, x_{\text{max}j}), x_{\text{min}j})$; else, $x'(i)_j = \min(\max(chld_2, x_{\text{max}j}), x_{\text{min}j})$.

(3) **Mutation:** The main way to produce new particles is through a mutation that can enhance the diversity of particles [26], the specific steps in Algorithm 5.
Algorithm 5: Mutation algorithm.

**Input:** (i) Recombination population \( A'' \).
(ii) Mutation parameter \( p_m \) and parameters \( \eta_m, r_1 \).
(iii) Lower bound of particles: \( x_{\text{min}} \), upper bound of particles: \( x_{\text{max}} \), and size of clone population: \( N' \).

**Output:** Mutation population \( A''' \).

**Step 1:** Generate a random number \( \text{rand} \in [0,1] \), if \( \text{rand} < p_m \),

**Step 1.1** If \( x''(i)_j - x_{\text{min}} < x_{\text{max}} - x''(i)_j \), then

\[
\delta = \frac{x''(i)_j - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}};
\]

**Step 1.2** Else \( \delta = \frac{x_{\text{max}} - x''(i)_j}{x_{\text{max}} - x_{\text{min}}} \).

**Step 2:** Calculate \( x_y = 1 - \delta \), and \( \text{val} = 2r_1 + (1 - 2r_1) \times (x_y^{\eta_m + 1}) \).

**Step 3:** Generate a random number \( \text{rand} \in [0,1] \), if \( \text{rand} < r_1 \), \( \delta_q = \text{val} \times \frac{1}{x_{\text{max}} - x_{\text{min}}} - 1 \); else, \( \delta_q = 1 - \text{val} \times \frac{1}{x_{\text{max}} - x_{\text{min}}} \).

**Step 4:** Calculate \( x'''(i)_j = x''(i)_j + \delta_q \times (x_{\text{max}} - x_{\text{min}}) \).

Then, the external archive will be updated by \( A''' \) and Algorithm 1. We can find that the clone operation can make particles inherit good information, the recombination operation can make good genes be inherited better, and the mutation operation can increase the diversity of particles. So, all operations make the algorithm more effective.

4.5. Chaotic Strategy

In order to enhance the diversity and local search ability of the algorithm in the late iterative stages, a local chaotic searching approach is used for the external archive, the specific steps in Algorithm 6.

Algorithm 6: Local chaotic algorithm.

**Input:** (i) External archive \( A \).
(ii) Maximum number of chaotic searching agents \( M' \).

**Output:** Clone population \( A_c \).

**Step 1:** Randomly generate a chaotic initial point \( y_0 = \text{rand} \), and chaotic searching number \( m = 1 \).

**Step 2:** According to the logistic map equation in chaotic systems, the following chaotic series is generated: \( y_j = 4y_j(1 - y_j) \).

**Step 3:** Renew the positions based on the following equation: \( x(i)_j = x(i)_j + \text{rand}(-1)^j / y_j \).

**Step 4:** If the termination criterion is satisfied, then output the \( A_c \). Otherwise, let \( m = m + 1 \) and return to **Step 2**.

4.6. Chaotic Multi-Objective Particle Swarm Optimization Algorithm Incorporating Clone Immunity (CICMOPSO)

**Step 1:** Initialize a population of particles \( X_N \), and each particle with random position vector \( x_i \) and velocity vector \( v_i \). Further initialize the maximum size of the external archive \( K \), maximum size of the active population \( NA \), maximum size of the clone population \( NC \), lower bound of particles \( x_{\text{min}} \), upper bound of particles \( x_{\text{max}} \), maximum number of generations \( T_{\text{max}} \), and the current generation number \( T = 0 \).

**Step 2:** Calculate the fitness of all particles in \( X_N(T) \).

**Step 3:** Initialize an update to the external archive \( A \).

**Step 4:** Calculate the parallel cell coordinates of particles with Equation (5).

**Step 5:** Calculate the Pareto entropy and difference entropy of particles with Equations (6) and (7).

**Step 6:** Evaluate the states of particles by the judgement condition.
Step 7: Adjust the inertia weight by the states of particles and calculate the learning factors.
Step 8: Select the global best solution with Algorithm 2.
Step 9: Renew the positions and velocities of the particles based on Equations (2) and (3).
Step 10: For the external archive, perform the clone operation (Algorithm 3), the recombination operation (Algorithm 4), and the mutation operation (Algorithm 5), then update to the external archive $A$.
Step 11: Conduct chaotic search for the external archive $A$ with Algorithm 6, obtain the chaotic population, then update to the external archive $A$.
Step 12: Calculate the fitness of the particles and renew every optimal position of the particles.
Step 13: Renew the external archive $A$ with Algorithm 1.
Step 14: If the termination criterion is satisfied, then output the external archive $A$. Otherwise, let $T = T + 1$ and return to Step 4.

4.7. Computational Complexity

In this section, the computational complexity of proposed algorithm is discussed. We consider the main steps in one generation in the main loop of algorithm. $M$ is the objective number, $N$ indicates the population size, $K$ is external archive size (in our algorithm, $K = N$)
(1) The parallel cell coordinate of particles computing Process requires $O(N \times M)$ time;
(2) The clone operation requires $O(N)$ time;
(3) The recombination operation requires $O(NC)$ time, $NC$ indicates the maximum size of Clone population;
(4) The mutation operation requires $O(NC)$ time;
(5) The chaotic operation requires $O(N)$ time;
(6) The external archive update process requires $O(N^2 \times M)$ time.

To summarize, the worst case overall computational complexity of CICMOPSOS when one generation is $O(N^2 \times M)$, which indicates that CICMOPSOS is computationally efficient.

5. Numerical Experimentation

5.1. Benchmark Function and Parameter Setting

In order to test the performance of the CICMOPSOS algorithm, the ZDT [27] (Table 2) and DTLZ [28] (Table 3) test functions were used in the experiments in this paper. We then compared the algorithm to NSGA-II [21], SPAE2 [29], and NICPSO [26]. For the parameters in this paper, the initial values of the learning factors were $c_1 = c_2 = 1.5$, the initial value of $w$ was 0.9, the recombination possibility $P_c = 0.8$, the mutation probability $P_m = 1/n$, which was the reciprocal of the decision variable dimension, the maximum number of the external archive was 100, the population size was 100, and maximum number of generations $T_{max} = 300$. Experiments were independently repeated 30 times. The optimization problems are described in Table 2.
Table 2. Benchmark function—ZDT.

<table>
<thead>
<tr>
<th>Name</th>
<th>Objective Functions</th>
<th>D</th>
<th>Variable Bounds</th>
</tr>
</thead>
</table>
| ZDT1  | \[
\begin{align*}
  f_1(x) &= x_1 \\
  f_2(x) &= g(x)(1 - \sqrt{\frac{x_1}{g(x)}}) \\
  g(x) &= 1 + 9 \sum_{i=2}^{n} \frac{x_i}{n-1} \\
  f_1(x) &= x_1
\end{align*}
\] | 30     | \(x_i \in [0, 1]\) |
| ZDT2  | \[
\begin{align*}
  f_2(x) &= g(x)[1 - (x_1/g(x))^2] \\
  g(x) &= 1 + 9 \sum_{i=2}^{n} x_i/(n-1) \\
  f_2(x) &= x_1
\end{align*}
\] | 30     | \(x_i \in [0, 1]\) |
| ZDT3  | \[
\begin{align*}
  f_2(x) &= g(x)(1 - \sqrt{x_1/g(x)} - x_1 \sin(10\pi x_1)/g(x)) \\
  g(x) &= 1 + 9 \sum_{i=2}^{n} x_i/(n-1) \\
  f_2(x) &= x_1
\end{align*}
\] | 30     | \(x_i \in [0, 1]\) |
| ZDT4  | \[
\begin{align*}
  f_2(x) &= g(x)(1 - \sqrt{x_1/g(x)}) \\
  g(x) &= 1 + 10(n - 1) + \sum_{i=2}^{n} [x_i^2 - 10\cos(4\pi x_i)] \\
  f_2(x) &= 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\
  f_2(x) &= g(x)[1 - (f_1(x)/g(x))^2] \\
  g(x) &= 1 + 9 \sum_{i=2}^{n} x_i/(n-1)^{0.25}
\end{align*}
\] | 10     | \(x_i \in [0, 1]\) \(x_i \in [-5, 5]\) |

Table 3. Benchmark function—DTLZ.

<table>
<thead>
<tr>
<th>Name</th>
<th>Objective Functions</th>
<th>D</th>
<th>Variable Bounds</th>
</tr>
</thead>
</table>
| DTL1  | \[
\begin{align*}
  f_1(x) &= 0.5(1 + g(x_M))\prod_{i=1}^{M-1} x_i \\
  f_{m=2,M-1}(x) &= 0.5(1 + g(x_M))\prod_{i=m-1}^{M-1} x_i \\
  f_M(x) &= 1 + g(x_M)\sin(\frac{\pi}{\sqrt{M}}) \\
  g(x) &= \sum_{x_i \in M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)) \\
  f_1(x) &= 0.5(1 + g(x_M))\prod_{i=1}^{M-1} -1\cos\left(\frac{\pi x_i}{\sqrt{M}}\right)
\end{align*}
\] | 30     | \(x_i \in [0, 1]\) |
| DTL2  | \[
\begin{align*}
  f_1(x) &= 0.5(1 + g(x_M))\prod_{i=1}^{M-1} -1\cos\left(\frac{\pi x_i}{\sqrt{M}}\right)
\end{align*}
\] | 30     | \(x_i \in [0, 1]\) |
| DTL3  | \[
\begin{align*}
  f_1(x) &= 0.5(1 + g(x_M))\prod_{i=1}^{M-1} -1\cos\left(\frac{\pi x_i}{\sqrt{M}}\right)
\end{align*}
\] | 30     | \(x_i \in [0, 1]\) |
| DTL4  | \[
\begin{align*}
  f_1(x) &= 0.5(1 + g(x_M))\prod_{i=1}^{M-1} -1\cos\left(\frac{\pi x_i}{\sqrt{M}}\right)
\end{align*}
\] | 30     | \(x_i \in [0, 1]\) |
| DTL5  | \[
\begin{align*}
  f_1(x) &= 0.5(1 + g(x_M))\prod_{i=1}^{M-1} -1\cos\left(\frac{\pi x_i}{\sqrt{M}}\right)
\end{align*}
\] | 30     | \(x_i \in [0, 1]\) |
| DTL6  | \[
\begin{align*}
  f_1(x) &= 0.5(1 + g(x_M))\prod_{i=1}^{M-1} -1\cos\left(\frac{\pi x_i}{\sqrt{M}}\right)
\end{align*}
\] | 30     | \(x_i \in [0, 1]\) |
5.2. Performance Indicators

(1) Convergence index [26]: Generational distance (GD) indicates the average distance between the obtained Pareto front and the true Pareto front. GD is defined as follow:

\[
GD = \frac{1}{k} \sum_{i=1}^{k} d_i
\]  

where \( d_i = \sqrt{\sum_{j=1}^{n} (PF_j - PF_{t,j})^2} \), \( k \) is the number of Pareto Solutions, \( n \) is the number of objective functions, \( PF_j \) indicate the \( j \)th obtained Pareto solution, \( PF_{t,j} \) indicate the nearest point on true Pareto front from \( PF_j \).

(2) Spacing index [26]: Spacing (S) specify the spread of the obtained Pareto front. It is defined as follow:

\[
S = \frac{1}{k-1} \sqrt{\sum_{p=2}^{k} (D_p - D)^2}
\]

where \( D_p = \sum_{i=1}^{n} |PF_{i,(p-1)} - PF_{i,p}| \), is the absolute difference between the two consecutive solutions in the obtained Pareto front \( PF \), \( D \) is the average of all \( D_p \).

5.3. Numerical Results

The convergence results are shown in Table 4. The CICMOPSO algorithm had better results than the NSGA-II and SPEA2 algorithms for all benchmark functions. For ZDT4, CICMOPSO was worse than NICPSO, but for other functions, CICMOPSO had the better performance. This finding indicates that the resulting Pareto fronts obtained by CICMOPSO are closer to the true Pareto fronts, and CICMOPSO can effectively improve convergence.

<table>
<thead>
<tr>
<th>Function</th>
<th>NSGA-II</th>
<th>SPEA2</th>
<th>NICPSO</th>
<th>CICMOPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZDT1</td>
<td>Aver</td>
<td>1.14 x 10^{-3}</td>
<td>3.82 x 10^{-3}</td>
<td>1.08 x 10^{-3}</td>
</tr>
<tr>
<td></td>
<td>Var</td>
<td>1.41 x 10^{-4}</td>
<td>4.91 x 10^{-3}</td>
<td>4.26 x 10^{-5}</td>
</tr>
<tr>
<td>ZDT2</td>
<td>Aver</td>
<td>4.88 x 10^{-1}</td>
<td>8.61 x 10^{-3}</td>
<td>7.74 x 10^{-4}</td>
</tr>
<tr>
<td></td>
<td>Var</td>
<td>2.77 x 10^{-2}</td>
<td>2.60 x 10^{-3}</td>
<td>2.99 x 10^{-5}</td>
</tr>
<tr>
<td>ZDT3</td>
<td>Aver</td>
<td>2.48 x 10^{-3}</td>
<td>9.72 x 10^{-3}</td>
<td>8.93 x 10^{-4}</td>
</tr>
<tr>
<td></td>
<td>Var</td>
<td>1.27 x 10^{-4}</td>
<td>5.23 x 10^{-3}</td>
<td>4.02 x 10^{-5}</td>
</tr>
<tr>
<td>ZDT4</td>
<td>Aver</td>
<td>5.13 x 10^{-1}</td>
<td>9.98</td>
<td>1.37 x 10^{-3}</td>
</tr>
<tr>
<td></td>
<td>Var</td>
<td>1.18 x 10^{-1}</td>
<td>2.01 x 10^{1}</td>
<td>7.42 x 10^{-3}</td>
</tr>
<tr>
<td>ZDT6</td>
<td>Aver</td>
<td>7.58 x 10^{-2}</td>
<td>1.93 x 10^{-2}</td>
<td>3.08 x 10^{-3}</td>
</tr>
<tr>
<td></td>
<td>Var</td>
<td>6.08 x 10^{-3}</td>
<td>1.40 x 10^{-3}</td>
<td>1.33 x 10^{-4}</td>
</tr>
</tbody>
</table>

From Table 5, we can see that NICPSO and CICMOPSO produced better results for the Spacing index (S) than NSGA-II and SPEA2. For ZDT1 and ZDT2, CICMOPSO has better performance than NICPSO, which has minimum mean and variance, but for other functions, NICPSO performed even better.
It is well known that multi-objective optimization problems, especially when the number of objectives ≥ 3, are more difficult to solve. Simulation results in Table 6 show that CICMOPSO can solve most objective function problems.

Table 6. Convergence index and Spacing index for the DTLZ model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Aver</th>
<th>Var</th>
<th>Aver</th>
<th>Var</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTLZ1</td>
<td>3.54 × 10⁻¹</td>
<td>2.88 × 10⁻¹</td>
<td>3.22 × 10⁻¹</td>
<td>2.76 × 10⁻¹</td>
</tr>
<tr>
<td>DTLZ2</td>
<td>1.93 × 10⁻¹</td>
<td>1.62 × 10⁻¹</td>
<td>7.40 × 10⁻¹</td>
<td>5.43 × 10⁻¹</td>
</tr>
<tr>
<td>DTLZ3</td>
<td>5.42</td>
<td>8.84</td>
<td>1.74</td>
<td>5.00</td>
</tr>
<tr>
<td>DTLZ4</td>
<td>1.95</td>
<td>6.15</td>
<td>9.17</td>
<td>1.21</td>
</tr>
<tr>
<td>DTLZ5</td>
<td>1.37 × 10⁻¹</td>
<td>8.29 × 10⁻²</td>
<td>4.51 × 10⁻¹</td>
<td>9.62 × 10⁻²</td>
</tr>
<tr>
<td>DTLZ6</td>
<td>1.45</td>
<td>1.07</td>
<td>9.29</td>
<td>6.69</td>
</tr>
</tbody>
</table>

6. Conclusions

A chaotic multi-objective particle swarm optimization algorithm incorporating clone immunity (CICMOPSO), which is used to solve multi-objective problems, was proposed in this paper. CICMOPSO uses the clone immunity strategy to maintain an external file and avoid the algorithm falling into a local optimal solution. Additionally, the Pareto entropy is used to dynamically adjust the algorithm parameters. The experimental results showed that CICMOPSO significantly outperforms all other algorithms based on the test problems with respect to two metrics. For problems with two-objective functions, CICMOPSO performed well, but for more-objective functions, the results are not satisfactory. We will improve CICMOPSO to make it suitable for solving more problems in the near future.

Author Contributions: All the authors have contributed equally to this paper. All the authors have read and approved the final manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (61561001) and First-Class Disciplines Foundation of NingXia (Grant No. NYLYXK2017B09).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

- PSO: Particle swarm optimization algorithm
- MOPSO: Multi-objective particle swarm
- GA: Genetic algorithm
DE differential evolution algorithm  
PCCS parallel cell coordinate system  
CICMOPSO chaos multi-objective particle swarm optimization incorporating clone immunity

References

9. Wang, S.; Zhang, Y.; Ji, G.; Yang, J.; Wu, J.; Wei, L. Fruit Classification by Wavelet-Entropy and Feedforward Neural Network Trained by Fitness-Scaled Chaotic ABC and Biogeography-Based Optimization. *Entropy* 2015, 17, 5711–5728. [CrossRef]


© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).