Abstract - An adaptive particle swarm optimization (PSO) on individual level is presented. By analyzing the social model of PSO, a replacing criterion based on the diversity of fitness between current particle and the best historical experience is introduced to maintain the social attribution of swarm adaptively by taking off inactive particles. The testing of three benchmark functions indicates it improves the average performance effectively. 

Key words: adaptive particle swarm optimization, evolutionary computation, social model

1. Introduction

Modern heuristic algorithms are considered as practical tools for nonlinear optimization problems, which do not require that the objective function to be differentiable or be continuous. The particle swarm optimization (PSO) algorithm [1] is an evolutionary computation technique, which is inspired by social behavior of swarms. It has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement [2].

Work presented in [3] describes the complex task of parameter selection in the PSO model. PSO has been proved to be a competitor to the standard genetic algorithm (GA). Comparisons between PSO and GA were done with regards to performance by Angeline [4], which points out that the PSO performs well in the early iterations, but has problems in reaching a near optimal solution in several benchmark functions.

To overcome this problem, some researchers have employed methods with adaptive parameters [5, 6]. One is deterministic mode, which the PSO parameters are changed according to the deterministic rules, such as a linear decreased inertia weight as the number of generation increasing [5], which are obtained according to the experience. The other is adaptive mode, which adjusts the parameters according to the feedback information, such as fuzzy adaptive inertia weight [6]. However, currently, we still have not captured the relations between different parameters and their effects toward different problem instances, such as dynamic optimization problems [2], due to the complex relationship among parameters.

Unlike the former efforts on adjusting PSO parameters, this paper will propose an efficient approach by adapting the swarm on individual level, which is realized by replacing the inactive particle with a fresh one in order to maintain the social attribution of swarm, according to the analyzing for the model of PSO. Both standard and adaptive versions are compared on three benchmark problems. The results suggest that the adaptive PSO enhance the performance effectively.

2. Standard particle swarm optimization (SPSO)

PSO is similar to the other evolutionary algorithms in that the system is initialized with a population of random solutions. Each potential solution, call particles, flies in the D-dimensional problem space with a velocity which is dynamically adjusted according to the flying experiences of its own and its colleagues. The location of the $i$th particle is represented as $X_i = (x_{1i}, \ldots,$
The process for implementing PSO is as follows:

a). Set current iteration generation \( G_i = 1 \). Initialize a population which including \( m \) particles. For the \( i \)th particle, it has random location \( X_i \) in specified space and for the \( d \)th dimension of \( V_{id} \), \( V_{id} = \text{Rand}_d() \times V_{max,d} \), where \( \text{Rand}_d() \) is a random value in the range [-1, 1];

b). Evaluate the fitness for each particle;

c). Compare the evaluated fitness value of each particle with its \( pbest \). If current value is better than \( pbest \), then set the current location as the \( pbest \) location. Furthermore, if current value is better than \( gbest \), then reset \( gbest \) to the current index in particle array;

d). Change the velocity and location of the particle according to the equations (1a) and (1b), respectively;

e). \( G_i = G_i + 1 \), loop to step b) until a stop criterion is met, usually a sufficiently good fitness value or \( G_e \) is achieve a predefined maximum generation \( G_{max} \).

The parameters of PSO includes: number of particles \( m \), inertia weight \( w \), acceleration constants \( c_1 \) and \( c_2 \), maximum velocity \( V_{max} \).

3. Adaptive particle swarm optimization (APSO)

As evolution goes on, the swarm might undergo an undesired process of diversity loss. Some particles become inactively while lost both of the global and local search capability in the next generations. For a particle, the lost of global search capability means that it will be only flying within a quite small space, which will be occurs when its location and \( pbest \) is close to \( gbest \) (if the \( gbest \) has not significant change) and its velocity is close to zero (for all dimensions) according to the equation (1); the lost of local search capability means that the possible flying cannot lead perceptible effect on its fitness. From the theory of self-organization [8], if the system is going to be in equilibrium, the evolution process will be stagnated. If \( gbest \) is located in a local optimum, then the swarm becomes premature convergence as all the particles become inactively.

To stimulate the swarm with sustainable development, the inactive particle should be replaced by a fresh one adaptively so as to keeping the non-linear relations of feedback in equation (1) efficiently by maintaining the social diversity of swarm.

However, it is hard to identify the inactive particles, since the local search capability of a particle is highly depended on the specific location in the complex fitness landscape for different problems.

Fortunately, the precision requirement for fitness value is more easily to be decided for specified problem. The adaptive PSO is executed by substituting the step d) of standard PSO process, as the pseudocode of adaptive PSO that is shown in Fig. 1. \( F_i \) is the fitness of the \( i \)th particle, \( F_{gbest} \) is the fitness of \( gbest \). \( \Delta F_i = f(F_i, F_{gbest}) \), where \( f(x) \) is a error function. The \( \epsilon \) is a predefined critical constant according to the precision requirement.
For the purpose of comparison, the asymmetric initialization method used in [4-6] is adopted here for population initialization. Table 1 lists the initialization ranges, and Table 2 lists the \( V_{max} \) and \( X_{max} \) values for all the functions, respectively. The acceleration constants are set as: \( c_1 = c_2 = 2 \). The fitness value is set as function value. We had 500 trial runs for every instance.

For APSO, \( \Delta F_i \) is set as a relative error function, which is \( (F_i - F_{gbest})/\min(\text{ABS}(F_i), \text{ABS}(F_{gbest})) \), where \( \text{ABS}(x) \) gets the absolute value of \( x \), \( \min(x_1, x_2) \) gets the minimum value between \( x_1 \) and \( x_2 \). The critical constant \( \epsilon \) is set as \( 1e-4 \), and the count constant \( T_c \) is set as \( 3 \).

In order to investigate whether the adaptive PSO scales well or not, different numbers of particles \( m \) are used for each function which different dimensions. The numbers of particles \( m \) are 20, 40, 80 and 160. \( G_{max} \) is set as 1000, 1500 and 2000 generations corresponding to the dimensions 10, 20 and 30, respectively.

**Table 1: Asymmetric initialization ranges**

<table>
<thead>
<tr>
<th>Function</th>
<th>Asymmetric Initialization Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>(15,30)</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>(2.56,5.12)</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>(300,600)</td>
</tr>
</tbody>
</table>

**Table 2: \( V_{max} \) and \( X_{max} \) values for each function**

<table>
<thead>
<tr>
<th>Function</th>
<th>( X_{max} )</th>
<th>( V_{max} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>600</td>
<td>600</td>
</tr>
</tbody>
</table>

**Table 3: The mean fitness values for the Rosenbrock function**

<table>
<thead>
<tr>
<th>( m )</th>
<th>( D )</th>
<th>( G_{max} )</th>
<th>( SPSO_{0.1} )</th>
<th>( SPSO_{0.5} )</th>
<th>( SPSO_{0.9} )</th>
<th>( APSO )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>1000</td>
<td>96.1715</td>
<td>66.01409</td>
<td>44.1374</td>
<td>28.8179</td>
</tr>
<tr>
<td>20</td>
<td>1500</td>
<td>214.6764</td>
<td>108.2865</td>
<td>87.2810</td>
<td>62.0391</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>2000</td>
<td>316.4468</td>
<td>183.8037</td>
<td>132.5973</td>
<td>95.5643</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>1000</td>
<td>70.2139</td>
<td>48.76523</td>
<td>24.3512</td>
<td>16.4028</td>
</tr>
<tr>
<td>40</td>
<td>1500</td>
<td>180.9671</td>
<td>63.88408</td>
<td>47.7243</td>
<td>37.1774</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>2000</td>
<td>299.7061</td>
<td>175.0093</td>
<td>66.6341</td>
<td>57.3000</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>10</td>
<td>1000</td>
<td>36.2945</td>
<td>15.81645</td>
<td>15.3883</td>
<td>11.6443</td>
</tr>
<tr>
<td>80</td>
<td>1500</td>
<td>87.2802</td>
<td>45.99998</td>
<td>40.6403</td>
<td>32.5682</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>2000</td>
<td>205.5596</td>
<td>124.4184</td>
<td>63.4453</td>
<td>55.2538</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>10</td>
<td>1000</td>
<td>24.4477</td>
<td>-</td>
<td>11.6283</td>
<td>6.9480</td>
</tr>
<tr>
<td>160</td>
<td>1500</td>
<td>72.8190</td>
<td>-</td>
<td>28.9142</td>
<td>25.9617</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>2000</td>
<td>131.5866</td>
<td>-</td>
<td>56.6689</td>
<td>45.6651</td>
<td></td>
</tr>
</tbody>
</table>

**FIG. 1** Inserted pseudocode of adaptive PSO

```
int[] similarCount = new int[m];  // at initialization stage
  // Next code is employed to replace step d)
  // in standard PSO process
  FOR (i=0; i<m; i++) {
    // for each particle
    IF (|i-g|= \&\& |\Delta F| \epsilon) {  // add 1
      similarCount[i] = 0;  // reset
    ELSE IF (similarCount[i] > T_c)  // predefined count
      THEN replace (the i-th particle);
    ELSE execute (step d) in standard PSO;
  }
```

4. Results and discussion

For comparison, three benchmark functions that are commonly used in the evolutionary computation literature [4-6] are used. All functions have same minimum value, which are equal to zero.

The function \( f_1 \) is the Rosenbrock function:

\[
f_1(x) = \sum_{d=1}^{D-1} (100(x_{d+1}^2 - x_d^2) + (x_d - 1)^2)
\]

(2a)

The function \( f_2 \) is the generalized Rastrigrin function:

\[
f_2(x) = \sum_{d=1}^{D} (x_d^2 - 10 \cos(2\pi x_d) + 10)
\]

(2b)

The function \( f_3 \) is the generalized Griewank function:

\[
f_3(x) = \frac{1}{4000} \sum_{d=1}^{D} x_d^2 - \prod_{d=1}^{D} \cos\left(\frac{x_d}{\sqrt{d}}\right) + 1
\]

(2c)

\( T_c \) is the count constant. The \texttt{replace()} function is employed to replace the \texttt{i}th particle, where the \( X_i \) and \( V_i \) is reinitialized by following the process in step a) of standard PSO, and its \( p_{best} \) is equal to \( X_i \).

The array \texttt{similarCount} is employed to store the counts which is satisfying the condition \( |\Delta F| < \epsilon \) in successively for the \texttt{ith} particle which is not \texttt{gbest}. The inactive particle is natural to satisfy the replace condition; however, if the particle is not inactively, it has less chance to be replaced as \( T_c \) increases.

For the purpose of comparison, the asymmetric initialization method used in [4-6] is adopted here for population initialization. Table 1 lists the initialization ranges, and Table 2 lists the \( V_{max} \) and \( X_{max} \) values for all the functions, respectively. The acceleration constants are set as: \( c_1 = c_2 = 2 \). The fitness value is set as function value. We had 500 trial runs for every instance.

For APSO, \( \Delta F_i \) is set as a relative error function, which is \( (F_i - F_{gbest})/\min(\text{ABS}(F_i), \text{ABS}(F_{gbest})) \), where \( \text{ABS}(x) \) gets the absolute value of \( x \), \( \min(x_1, x_2) \) gets the minimum value between \( x_1 \) and \( x_2 \). The critical constant \( \epsilon \) is set as \( 1e-4 \), and the count constant \( T_c \) is set as 3.

In order to investigate whether the adaptive PSO scales well or not, different numbers of particles \( m \) are used for each function which different dimensions. The numbers of particles \( m \) are 20, 40, 80 and 160. \( G_{max} \) is set as 1000, 1500 and 2000 generations corresponding to the dimensions 10, 20 and 30, respectively.
is decided by the precision almost all cases. For example, as PSO has the capacity to achieve high performance.

PSO in [6] with fuzzy adaptive have better results than all of other PSO version for w=0.4. As an version forיאו, and PSO[6] is the results of standard PSO with w=0.4 as an version for comparison. APSO is the results of adaptive PSO in this work with w=0.4.

By compare the results, it is easy to see that APSO have better results than all of other PSO version for almost all cases. For example, as SP50 w=0.4 get worst solutions due to premature convergence for Rastrigrin function, the adaptive version APSO has the capacity to sustainable development. It means that the adaptive PSO has the capacity to achieve high performance.

5. Conclusion

In this paper, an adaptive particle swarm optimizer was introduced to improve the performance. The adaptive criterion is appended on individual level. Since the critical constant ε is decided by the precision requirement to fitness, it is more easily to be decided for different problems. Three benchmark functions have been used for testing. The simulation results illustrate the performance of adaptive PSO can improve the performance. This adaptive method may be also used for other evolutionary computation technologies, such as genetic algorithms.

Table 3 to 5 lists the mean fitness values for the three benchmark functions. Where PSO is the results of standard PSO in [5] with a linearly decreasing w which from 0.9 to 0.4, FPSO is the results of fuzzy adaptive PSO in [6] with fuzzy adaptive w, and SPSO.4 is the results of standard PSO with w=0.4 as an version for comparison. APSO is the results of adaptive PSO in this work with w=0.4.

By compare the results, it is easy to see that APSO have better results than all of other PSO version for almost all cases. For example, as SP50 w=0.4 get worst solutions due to premature convergence for Rastrigrin function, the adaptive version APSO has the capacity to sustainable development. It means that the adaptive PSO has the capacity to achieve high performance.

References