A dynamic inertia weight particle swarm optimization algorithm

Bin Jiao a,b,*, Zhigang Lian a, Xingsheng Gu a

a Research Institute of Automation, East China University of Science and Technology, Shanghai 200237, China
b Electrical Engineering Department, Shanghai DianJi University, Shanghai 200240, China

Accepted 13 September 2006

Abstract

Particle swarm optimization (PSO) algorithm has been developing rapidly and has been applied widely since it was introduced, as it is easily understood and realized. This paper presents an improved particle swarm optimization algorithm (IPSO) to improve the performance of standard PSO, which uses the dynamic inertia weight that decreases according to iterative generation increasing. It is tested with a set of 6 benchmark functions with 30, 50 and 150 different dimensions and compared with standard PSO. Experimental results indicate that the IPSO improves the search performance on the benchmark functions significantly.

© 2006 Elsevier Ltd. All rights reserved.

1. Introduction

The particle swarm optimization (PSO) is an evolutionary computation technique developed by Dr. Eberhart and Dr. Kennedy in 1995 [2,8], inspired by social behavior of bird flocking or fish schooling. PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas where GA can be applied. Recently, several investigations have been undertaken to improve the performance of standard PSO and have obtained rich harvests. Clerc and Kennedy in [1] have reached the particle swarm explosion stability and convergence in a multi-dimensional complex space. Eberhart and Shi have compared the genetic algorithms with PSO in [3] and have investigated PSO developments, applications and resources in [4] and have presented a modified particle swarms optimizer in [16,17]. Fan has presented a modification to PSO algorithm in [5] and He et al. have put forward a particle swarm optimizer with passive congregation in [6]. Liu et al. presented an improved particle swarm optimization combined with chaos in [13]. Shi et al. in [19] have presented an improved GA and a novel PSO-GA-based hybrid algorithm. Kennedy in [10] has studied the PSO social adaptation of knowledge. Robinson et al. in [15] have investigated particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. Shi and Eberhart have researched parameter selection in particle swarm optimization, evolutionary

* Corresponding author. Address: Research Institute of Automation, East China University of Science and Technology, Shanghai 200237, China.
E-mail address: binjiaocn@163.com (B. Jiao).

0960-0779/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
programming in [18]. Kennedy and Mendes in [11] investigated the impacts of population structures to the search performance of SPSO. Other investigations on improving PSO’s performance were undertaken using cluster analysis [9]. Trelea in [20] has researched the PSO algorithm convergence analysis and parameter selection. Addition in papers [7,14] authors researched the chaotic optimisation and their application. We in paper [12] presented a novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan, which is more effective than GA.

In this paper we propose a dynamic inertia weight PSO algorithm for optimization problems. This work differs from the existing ones at least in two aspects: firstly, it proposes a dynamic inertia weight PSO algorithm iterative formula, which uses the dynamic inertia weight that decreases according to iterative generation increasing. The second is to compare the standard PSO with IPSO and obtain that IPSO is more efficacious for optimization problem. The rest of the paper is organized as follow: The next section introduces the standard PSO. A dynamic inertia weight particle swarm optimization algorithm is presented in Section 3. In Section 4, we describe the test functions, experimental settings, and compare experimental results of PSO with IPSO. Finally, Section 5 summarizes the contribution of this paper and conclusions.

2. Standard particle swarm optimizer

PSO that learned from the scenario is an evolutionary computation technique mimicking the behavior of flying birds and their means of information exchange and developed by Dr. Eberhart and Dr. Kennedy. In PSO, each single solution is a “bird” in the search space. We call it “particle”. All of particles have fitness values that are evaluated by the fitness function to be optimized, and have velocities that direct the flying of the particles. Similar to other population-based algorithms, PSO as an optimization tool can solve a variety of difficult optimization problems. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. Compared with GA, the advantages of PSO are that PSO is easy to implement and there are fewer parameters to adjust.

The population of PSO is called a swarm and each individual in the population of PSO is called a particle. The i\textsuperscript{th} particle at iteration k has the following two attributes [2,6,8]:

(a) A current position in an N-dimensional search space \(X_i^k = (x_i^1, \ldots, x_i^N)\), where \(x_i^k \in [l_n, u_n]\), \(1 \leq n \leq N\), \(l_n\) and \(u_n\) is lower and upper bound for the nth dimension, respectively.

(b) A current velocity \(V_i^k = (v_i^1, \ldots, v_i^N)\) which is bounded by a maximum velocity \(V_{\text{max}}^k = (v_{\text{max},1}, \ldots, v_{\text{max},N})\) and a minimum velocity \(V_{\text{min}}^k = (v_{\text{min},1}, \ldots, v_{\text{min},N})\).

In every search-iteration, each particle is updated by following two “best” values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another “best” value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest. After finding the two best values, the particle updates its velocity and positions with following formulas [2,8]:

\[
\begin{align*}
V_i(k + 1) &= wV_i(k) + c_1r_1(P_i(k) - X_i(k)) + c_2r_2(P_g(k) - X_i(k)) \\
X_i(k + 1) &= X_i(k) + V_i(k + 1)
\end{align*}
\]

where \(P_i\) is the best previous position of the i\textsuperscript{th} particle (also known as pbest). According to the different definitions of \(P_g\), there are two different versions of PSO. If \(P_g\) is the best position among all the particles in the swarm (also known as gbest), such a version is called the global version. If \(P_g\) is taken from some smaller number of adjacent particles of the population (also known as lbest), such a version is called the local version. \(P_i\) and \(P_g\) are given by the following equations, respectively:

\[
\begin{align*}
P_i &= \left\{ P_i : f(X_i) \geq f(P_i) \right\} \\
P_l &= \left\{ P_l : f(X_l) < f(P_l) \right\}
\end{align*}
\]

\[
P_g \subseteq \{ P_0, P_1, \ldots, P_m \} \text{ s.t. } \min(f(P_0), f(P_1), \ldots, f(P_m))
\]

where \(f\) is the objective function, \(m \leq M\) and \(M\) is the total number of particles. In (1) and (2) \(k\) represents the iterative number, and the variables \(c_1, c_2\) are learning factors, usually \(c_1 = c_2 = 2\), which control how far a particle will move in a
single iteration. \( r_1, r_2 \) are elements from two uniform random sequences in the range \((0, 1)\); \( r_1 \sim U(0,1) \); \( r_2 \sim U(0,1) \) and \( w \) is an inertia weight in \([16]\), which is initialized typically in the range of \([0,1]\). A larger inertia weight facilitates global exploration and a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area \([17]\). The termination criterion for the iterations is determined according to whether the max generation or a designated value of the fitness of \( P_g \) is reached.

3. Dynamic inertia weight particle swarm optimization algorithm

The foundation of PSO is based on the hypothesis that social sharing of information among conspecifics offers an evolutionary advantage. The standard PSO model is based on the following two factors: (1) The autobiographical memory, which remembers the best previous position of each individual \( P_i \) in the swarm; (2) The publicized knowledge, which is the best solution \( P_g \) found currently by the population. Therefore, the sharing of information among conspecifics is achieved by employing the publicly available information \( P_g \). There is no information sharing among individuals except that \( P_g \) broadcasts the information to the other individuals. Therefore, the population may lose diversity and is more likely to confine the search around local minima if committed too early in the search to the global best found so far.

3.1. Description of dynamic inertia weight

In standard PSO algorithm, the information of individual best and global best were shared by next generation particles. In this paper we present an improved PSO algorithm, which uses the dynamic inertia weight that decreases according to iterative generation increasing. The detailed information will be given in following.

Suppose that the searching space is \( D \)-dimensional and \( m \) particles form the colony. The \( i \)th particle represents a \( D \)-dimensional vector \( X_i \) \((i = 1, 2, \ldots , m)\). It means that the \( i \)th particle locates at \( X_i = (x_{i1}, x_{i2}, \ldots , x_{id}) \) \((i = 1, 2, \ldots , m)\) in the searching space. The position of each particle is a potential result. We could calculate the particle’s fitness by putting its position into a designated objective function. When the fitness is lower, the corresponding \( P_i \) is updated. The \( i \)th particle’s “flying” velocity is also a \( D \)-dimensional vector, denoted as \( V_i = (v_{i1}, v_{i2}, \ldots , v_{id}) \). Denote the best position of the \( i \)th particle as \( P_i = (p_{i1}, p_{i2}, \ldots , p_{id}) \) and the best position of the colony as \( P_g = (p_1, p_2, \ldots , p_D) \) respectively.

\[
V_i(k + 1) = w'V_i(k) + c_1r_1(P_i(k) - X_i(k)) + c_2r_2(P_g(k) - X_i(k))
\]
\[
X_i(k + 1) = X_i(k) + V_i(k + 1)
\]

where \( w' = w*u^{-k} \) \((w \in [0, 1], u \in [1.0001, 1.005])\), \( w' \) is an inertia weight \([16]\), which is initialized typically in the range of \([0,1]\). A larger inertia weight facilitates global exploration and a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area \([18]\). The variables \( c1 \) and \( c2 \) are acceleration constants \([4]\), which control how far a particle will move in a single iteration, and are same as in (1) and (2).

3.2. Step of IPSO

Step 1: Let initialization iterative number be \( k = 0 \), initialization population size be \( \text{psize} \), the termination iterative number be \( \text{Maxgen} \). Give birth to psize initializing particles. Calculate each particle’s fitness value of initialization population, and let first generation \( P_i \) be initialization particles, and choose the particle with the best fitness value of all the particles as the \( P_g \) (gBest).

Step 2: Using (5) and (6) gives birth to the next generational particles \( X_i \). If the fitness value is better than the best fitness value \( P_i(k) \) \((p\text{Best})\) in history, let current value as the new \( P_i(k) \). Choose the particle with the best fitness value of all particles as the \( P_g(k) \). If \( k == \text{Maxgen} \), go to Step 3, or else let \( k = k + 1 \) go to Step 2.

Step 3: Put out the \( p_g \).

The searching is a repeat process, and the stop criteria are that the maximum iteration number is reached or the minimum error condition is satisfied. The stop condition depends on the problem to be optimized. In IPSO algorithm, each particle of the swarm shares mutual information globally and benefits from discoveries and previous experiences of all other colleagues during the search process.
4. Numerical simulation

4.1. Test functions

To illustrate the effectiveness and performance of IPSO algorithm for optimization problems, a set of 6 representative benchmark functions was employed to evaluate it in comparison with standard PSO. Many authors tested algorithm using them widely.

**Sphere model:** \(f_1(x) = \sum_{i=1}^{30} x_i^2\).

**Schwefel’s Problem 1.2:** \(f_2(x) = \sum_{i=1}^{30} (\sum_{j=1}^{i} x_j)^2\)

**Generalized Rosenbrock’s function:** \(f_3(x) = \sum_{i=1}^{29} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)\).

**Ackley’s function:**
\[
\begin{align*}
f_4(x) &= -20 \exp \left( -0.2 \sqrt{\frac{1}{30} \sum_{i=1}^{30} x_i^2} \right) - \exp \left( \frac{1}{30} \sum_{i=1}^{30} \cos 2\pi x_i \right) + 20 + e
\end{align*}
\]

**Generalized Griewank function:**
\[
\begin{align*}
f_5(x) &= \frac{1}{4000} \sum_{i=1}^{30} (x_i - 100)^2 - \prod_{i=1}^{30} \cos \left( \frac{x_i - 100}{\sqrt{i}} \right) + 1.
\end{align*}
\]

**Generalized Penalized functions:**
\[
\begin{align*}
f_6(x) &= \frac{\pi}{30} \left( 10 \sin^2(\pi y_1) + \sum_{j=1}^{n} (y_j - 1)^2 \times [1 + 10 \sin^2(\pi y_{j+1})] \right) + \sum_{i=1}^{30} u(x_i, 10, 100, 4)
\end{align*}
\]

where
\[
u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \leq x_i \leq a \\ k(-x_i - a)^m, x_i < -a \end{cases}
\]

They can be grouped as unimodal (function \(f_1-f_3\)) and multimodal functions (function \(f_4-f_6\)) where the number of local minima increases exponentially with the problem dimension.

4.2. Experimental results and comparison

The experimental results (i.e., the mean and the standard deviations of the function values found in 10 runs) for each algorithm on each test function are listed in Table 1. To evaluate the performance of the proposed IPSO, global version of standard PSO is used for comparison. All experiments were repeated for 10 runs. The parameters of PSO and IPSO algorithm were listed in Table 1 too.

**Remark.** In Table 1, \(F\) and \(n\) denote function and its dimension respectively; FSS denotes the feasible solution space, and OS is the optimal solution; PG indicates the population size and algorithm terminate generation. In addition, computing experiments the \(u\) equal to 1.00002.

To get the average performance of the IPSO algorithm, ten runs on each problem instance were performed and the solution quality was averaged. The best average solutions found by IPSO are illustrated with italic letters and the best solutions found are illustrated with bold letters. From simulation we can obtain that the IPSO algorithm is clearly better than the standard PSO for optimization problem. From Table 1 one can observe that \(w \in [0.3, 0.4]\) have the highest performance since using them has smaller arithmetic mean and smaller lower bound in relation to the solutions obtained by the other \(w\) value. The IPSO is efficacious especially for middle and large size optimisation problems, and its superiority is shown adequately in function \(f_1\) and \(f_4\) with dimension 150 and 100, respectively. Maybe the IPSO algorithm is not more efficacious than standard PSO for some functions with small size, but it is clearly more effective for same problem with middle or large size. The convergence rate figures of most effective IPSO compare with standard PSO for 6 instances are as following. IPSO and SPSO for \(f_6\) with dimension 100.

**Convergence rate figure of IPSO(\(\mu = 1.00002, w = 0.3\) comparing with standard PSO for function**

\[f_1 - f_6\]
Table 1  
Comparison results of the PSO algorithm and the IPSO algorithm  

<table>
<thead>
<tr>
<th>$F$</th>
<th>$n$</th>
<th>FSS</th>
<th>OS</th>
<th>PG</th>
<th>Minimum/average</th>
<th>$w$</th>
<th>PSO</th>
<th>IPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>−100</td>
<td>0</td>
<td>200,100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>−100</td>
<td>0</td>
<td>200,3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>−32</td>
<td>0</td>
<td>200,1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>−600</td>
<td>0</td>
<td>200,1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>−50</td>
<td>0</td>
<td>200,1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_6$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>−50</td>
<td>0</td>
<td>100,1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
From above figure we can discover that the convergence rate of dynamic inertia weight PSO is clearly faster than the standard PSO on the benchmark functions. At the same time, the best solution get by dynamic inertia weight PSO is more optimum than by standard PSO.
5. Conclusions and perspectives

In this paper we have discussed a dynamic inertia weight PSO algorithm for optimization problem. Although it does not guarantee the optimality for some instances, such an approach provides solutions with good quality in a reasonable time limit, and compared with standard PSO, it is more efficacious. The performance of the improved approach is evaluated in comparison with the results obtained from standard PSO for six representative instances and obtained results show the effectiveness of the proposed approach. The proposed dynamic inertia weight PSO algorithm approach in this paper can be considered as effective mechanisms from this point of view.

There are a number of research directions that can be considered as useful extensions of this research. Although the proposed algorithm is tested with six representative instances, a more comprehensive computational study should be made to test the efficiency of proposed solution technique. For parameters of dynamic inertia weight PSO algorithm, in this paper we just studied the different w affection, so the various $\mu$ effect must be researched in future work. In the future the dynamic inertia weight PSO algorithm should be used for solving other discrete combinatorial optimization problems such as FSSP, JSSP, TSP etc. The development of PSO is still ongoing. And there are still many unknown areas in dynamic inertia weight PSO research such as the mathematical validation of particle swarm theory.

Acknowledgements

This work is supported by The National Natural Science Foundation of China (Grant No. 60274043), the Key Technologies Program of Shanghai Municipal Science and Technology Commission (Grant No. 04dz11008) and the Key Technologies Program of Shanghai educational committee (Grant No. 05ZZ73) (2005).

References


