Information Sciences 274 (2014) 70–94

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Improving particle swarm optimization using multi-layer searching strategy

Lin Wang, Bo Yang *, Yuehui Chen

Shandong Provincial Key Laboratory of Network based Intelligent Computing, University of Jinan, Jinan 250022, China

ARTICLE INFO

Article history:
Received 11 March 2012
Received in revised form 19 February 2014
Accepted 28 February 2014
Available online 10 March 2014

Keywords:
Particle swarm optimization
Multi-layer particle swarm optimization
Evolutionary computation
Swarm intelligence

ABSTRACT

In recent years, particle swarm optimization (PSO) algorithm has been used to solve global optimization problems. This algorithm is widely used as an effective optimization tool in various applications. However, traditional PSO consists of only two searching layers and thus often results in premature convergence into the local minima. Thus, multi-layer particle swarm optimization (MLPSO) is proposed in this paper to improve the performance of traditional PSO by increasing the two layers of swarms to multiple layers. The MLPSO strategy increases the diversity of searching swarms to improve its performance when solving complex problems. The experiment indicates that the novel approach improves the final results and the convergence speed.

1. Introduction

Traditional optimization algorithms cannot solve highly complex real-world problems because of their inflexible structure resulting from incomplete or noisy data and other multi-dimensional problems. A suitable solution to such problems is the nature-inspired optimization algorithm developed in recent years. One such algorithm is the particle swarm optimization (PSO) algorithm, which is based on the foraging of birds. This algorithm has been widely utilized as an optimization tool in various applications such as communication, finance, energy, medicine, materials science, and remote sensing [7,13,17,28,33,46,50].

Traditional PSO consists of only two searching layers: one for searching the personal best solution and the other for searching the global best solution. All of the particles fly in the solution space following the two *bests*. However, the global best solution can easily dominate all the particles in multi-modal functions and influence a particle to move toward its direction, even if the particle is in a local optimum region far from the global optimum. If the personal and global best solutions are in the same region of the particle's current position and if the particle is oriented toward a local optimum, traditional PSO may be unable to jump out of the local optimum area.

The question that arises here is whether we can use more than two layers to search multi-modal regions thoroughly and avoid lapsing into the local optimum?

To address this issue, we propose a multi-layer PSO (MLPSO), which increases the swarm layers from two to multiple layers. In the MLPSO strategy, the best positions are located simultaneously in a number of potential optimum regions at each layer, and the upper layer leads the lower layer in thoroughly searching the multi-modal regions. In this manner, the MLPSO can jump out of the local optimum via the cross-layered cooperative behavior of the whole swarm.

* Corresponding author. Tel.: +86 53182765717; fax: +86 53187968014.
E-mail address: yangbo@ujn.edu.cn (B. Yang).

http://dx.doi.org/10.1016/j.ins.2014.02.143
0020-0255/

The rest of the paper is organized as follows. Section 2 reviews PSO. Section 3 describes the details and analyzes MLPSO. Section 4 outlines and discusses the experimental results. Section 5 concludes the study.

2. Particle swarm optimization

2.1. Related works

PSO algorithm is a global numerical algorithm based on the foraging of birds. This algorithm was proposed by Kennedy and Eberhart [20] in 1995. In this algorithm, each solution to a certain problem is represented by a particle in the search space. All of these particles form a population. The velocity of a particle decides its flying direction and distance. The particles fly in the solution space following the best solution. After a certain number of iterations, the position vector of the best particle serves as the approximate optimum solution to the problem.

PSO emulates the swarm behavior of birds, and the particles represent points in the d-dimensional search space. This algorithm first initializes the population with a group of random particles and then iteratively searches for the optimum solution. The particles update themselves by tracking two vectors: the best solution found by the particle itself *pbest* and the best solution found by the population *gbest*. In each iteration step, after these two vectors are updated, the particle updates its new velocity and position of the ith dimension as follows:

$$\begin{cases} v_{k+1}^i = v_k^i + \varphi_1 r(pbest_k^i - x_k^i) + \varphi_2 r(gbest_k^i - x_k^i) \\ x_{k+1}^i = x_k^i + v_{k+1}^i \end{cases} \tag{1}$$

where k is the iteration number and x is the particle position (i.e., a possible solution to the problem). v is the particle velocity vector and r represents random positive numbers drawn from a uniform distribution $[0,1]$. φ_1 and φ_2 are acceleration constants. *pbest* is the best previous position that yields the best fitness value for the particle and *gbest* is the best new position discovered by the whole population. The complete algorithm of the traditional PSO is shown in Algorithm 1.

Algorithm 1. Algorithm of Particle Swarm Optimization.

Input: Population size, acceleration constants φ_1, φ_2 and maximum velocity *VMAX*.
Output: The best solution.
Initialize position x, velocity v for all of particles. The best position *pbest* and *gbest* are also initialized
k = 0
While *termination condition has not met* **do**
  Evaluate the fitness for each particle according to its position vector x;
  Update the best position *pbest* for each particle;
  Update the best position *gbest* for the whole population;
  for *id*=1 **to** *population size* **do**
    for *i*=1 **to** *d* **do**
      $v^i = v^i + \varphi_1 r(pbest^i - x^i) + \varphi_2 r(gbest^i - x^i)$;
      $v^i = min(VMAX^i, max(-VMAX^i, v^i))$;
      $x^i = x^i + v^i$;
    end
  end
end
Return the best position.

Numerous attempts have been made to restrict the velocity value with the use of additional strategies [40–42,10,5,6,11,21,38,51]. Shi and Eberhart [40] introduced inertia weight. They proposed a linearly decreasing inertia weight with iterative generations. The maximal and minimal weights are usually set to 0.9 and 0.4, respectively [40,41]. Aside from the inertia weight and the constriction factor, acceleration coefficients are also important parameters in PSO. Kennedy and Eberhart [20] suggested a fixed value of 2.0, which has been adopted by numerous researchers. Ratnaweera et al. [38] proposed a PSO algorithm with linearly time-varying acceleration coefficients, where a large φ_1 and a small φ_2 are set at the beginning and are gradually reversed during the search. Zhan et al. [51] proposed an adaptive PSO that enables the automatic control of inertia weight, acceleration coefficients, and other algorithmic parameters at run time to improve search efficiency and convergence speed. Elitist learning strategy is then performed, during which the evolutionary state is classified as convergence state.

Other researchers have attempted to integrate mechanisms of different optimization algorithms to PSO [27,2,15,48,8,14,31,12,4]. Lovbjerg et al. [27] discussed the integration of the strategies of evolutionary computation with PSO. Instead of fitness, subpopulations and breeding probability fitness are introduced to PSO. Angeline [2] improved PSO