Applied Soft Computing 28 (2015) 138–149

Contents lists available at ScienceDirect

Applied Soft Computing

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

A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques

Limin Zhang[a,b,c,*], Yinggan Tang[a], Changchun Hua[a], Xinping Guan[a,b]

[a] Institute of Electrical Engineering, Yanshan University, Qinhuangdao 066004, China
[b] Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China
[c] Department of Mathematics and Computer Science, Hengshui University, Hengshui 053000, China

ARTICLE INFO

Article history:
Received 1 April 2014
Received in revised form 15 October 2014
Accepted 20 November 2014
Available online 9 December 2014

Keywords:
Particle swarm optimization
Monte Carlo
Gaussian distribution
Bayesian techniques

ABSTRACT

Particle swarm optimization is a stochastic population-based algorithm based on social interaction of bird flocking or fish schooling. In this paper, a new adaptive inertia weight adjusting approach is proposed based on Bayesian techniques in PSO, which is used to set up a sound tradeoff between the exploration and exploitation characteristics. It applies the Bayesian techniques to enhance the PSO's searching ability in the exploitation of past particle positions and uses the cauchy mutation for exploring the better solution. A suite of benchmark functions are employed to test the performance of the proposed method. The results demonstrate that the new method exhibits higher accuracy and faster convergence rate than other inertia weight adjusting methods in multimodal and unimodal functions. Furthermore, to show the generalization ability of BPSO method, it is compared with other types of improved PSO algorithms, which also performs well.

1. Introduction

Particle swarm optimization (PSO) was firstly introduced by Kennedy and Eberhart in 1995 [1]. It belongs to evolutionary algorithm (EA), however differs from other evolutionary algorithms, which is inspired by the emergent motion of a flock of birds searching for food. PSO performs well in finding good solutions for optimization problems [2], and it has become another powerful tool besides other evolutionary algorithms such as genetic algorithms (GA) [3]. PSO is initialized with a population of particles randomly positioned in an n-dimensional search space. Every particle in the population has two vectors, i.e., velocity vector and position vector. The PSO algorithm is recursive, which motivates social search behavior among particles in the search space, where every particle represents one point. In comparison with other EAs such as GAs, the PSO has better search performance with faster and more stable convergence rates.

Maintaining the balance between global and local search in the course of all runs is critical to the success of an optimization algorithm [4]. All of the evolutionary algorithms use various methods to achieve this goal. To bring about a balance between the two searches, Shi and Eberhart proposed a PSO based on inertia weight in which the velocity of each particle is updated according to a fixed equation [5]. A higher value of the inertia weight implies larger incremental changes in velocity, which means the particles have more chances to explore new search areas. However, smaller inertia weight means less variation in velocity and slower updating for particle in local search areas.

In this paper, the inertia weight strategies are categorized into three classes. The first class is simple that the value of the inertia weight is constant during the search or is selected randomly. In [6], the impact of the inertia weight is analyzed on the performance of the PSO. In [7], Eberhart and Shi use random value of inertia weight to enable the PSO to track the optima in a dynamic environment. In the second class, the inertia weight changes with time or iteration number. We name the strategy as time-varying inertia weight strategy. In [8,9], a linear decreasing inertia weight strategy is introduced, which performs well in improving the fine-tuning characteristic of the PSO. Lei et al. use the Sugeno function as inertia weight declined curve in [10]. Many other similar linear approaches and nonlinear methods are applied in inertia weight strategies such as in [11–13]. The last class is some methods that inertia weight is revised using a feedback parameter. In [4], a fuzzy system is proposed to dynamically adapt the inertia weight. In [14], the inertia weight is determined by the ratio of the global best fitness and the average of particles' local best fitness in each iteration. In [15], A new strategy is presented that the inertia weight is dynamically

* Corresponding author at: Institute of Electrical Engineering, Yanshan University, Qinhuangdao 066004, China. Tel.: +86 18230357919.
E-mail address: limin_zhang@yeah.net (L. Zhang).

http://dx.doi.org/10.1016/j.asoc.2014.11.018
1568-4946/

adjusted according to average absolute value of velocity, which follows a given nonlinear ideal velocity by feedback control, which can avoid the velocity closed to zero at the early stage. In [16], dynamic acceleration parameters (DAP) method is proposed, which owns a mechanism to self-tune the acceleration parameters by utilising the averaged velocity information of the particles. In [17], a new adaptive inertia weight strategy is proposed based on the success rate of the particles. In this strategy the success rate of the particles is used as a feedback parameter to realize the state of the particles in the search space and hence to adjust the value of inertia weight.

A series of other studies based on mutation strategy has been done on the analysis and development of the PSO since it was introduced in 1995. These approaches aim to improve the PSO's convergence velocity. In [18], a mutation operator is used that changes a particle dimension value using a random number drawn from a Gaussian distribution (GPSO). A particle is selected for mutation using a mutation rate that is linearly decreased during a run. In [19], a mutation strategy is proposed that a particle position is changed using a random number drawn from a Gaussian distribution. A mutation operator in [20] is similar to that of Ref. [18], but a Cauchy probability distribution is used instead (CPSO). The Cauchy distribution curve is similar to the Gaussian distribution curve, except it has more probability in its tails and thus making it more likely to return larger values. In [21], the HPSO with a wavelet mutation (HWPSO) is proposed, in which the mutation incorporates with a wavelet function.

Hybrid PSOs (HPSOs) have been proposed to enhance the performance of the PSO, in which different mutation strategies are used. In [22], premature convergence is avoided by adding a mutation strategy, i.e., a perturbation to a randomly selected particle's velocity vector. A Cauchy mutation (HPSO) [23] is proposed, which is used in best particle mutation, so that the best particle could lead the rest of the particles to better positions. The algorithm in [24] called IPSO + ACJ is tested on a suite of well known benchmark multimodel functions and the results. The main idea of our new jump strategy is that pbest and gbest are selected as mutated particles when they have not been improved in a predefined number of iterations. In [25], Ant colony optimization (ACO) and PSO work separately at each iteration and produce their solutions. In [26], The new mutation strategy makes it easier for particles in hybrid MRPSO (HMRPSO) to find the global optimum and seek a balance between the exploration of new regions and the exploitation of the old regions.

Fuzzy approaches for PSO is a hot topic in these years. In [27], a fuzzy system is used to dynamically adjust the inertia weight and learning factors of PSO in each topology. In [28], a dynamic parameter adaptation is proposed to improve the convergence and diversity of the swarm in PSO using fuzzy logic. Valdezis et al. [29] introduced an improved FPSO + FGA hybrid method, which combines the advantages of PSO and GA. See Ref. [30] for a comprehensive review on the application of fuzzy logic in PSO, ACO and Gravitational Search Algorithm (GSA).

There are some works that have used the Bayesian technique in smart computing. In [31], the Dynamic Bayesian Network (DBN) is used in PSO algorithm for particle motion. In swarm optimization, each particle tries to track the trajectory toward the place of better fitness. Martens et al. [32] explained scientifically the application of the Bayesian network in ACO. In our paper, Bayesian techniques is introduced into the PSO algorithm with a view to enhance its adaptive search ability. We call this algorithm as PSO with Bayesian techniques (BPSO). Different from other inertia weight strategies in the references, BPSO algorithm adjusts the inertia weight ω based on the past particle places automatically. This new development gives particles more opportunity to explore the solution space than a standard PSO. Furthermore, the new algorithm accelerates search velocity for the particles in valuable search-space regions.

This paper is organized as follows. In Section 2, generic PSO theory is reviewed and the change of particle position ε is analyzed. Section 3 introduces the advantage of PSO with Bayesian Techniques. Section 4 shows the experimental settings for the benchmarks, simulation results and parameters analysis in the BPSO. Finally, Section 5 presents conclusions resulting from the study.

2. Particle swarm optimization

2.1. Generic PSO theory

PSO is also a population-based stochastic optimization algorithm and starts with an initial population of randomly generated solutions called particles. Each particle in PSO has a position and a velocity. PSO remembers both the best position found by all particles and the best positions found by each particle in the search process. For a search problem in an n-dimensional space, a potential solution is represented by a particle that adjusts its position and velocity according to Eqs. (1) and (2):

$$v_{i,d}(t+1) = \omega v_{i,d}(t) + c_1 r_1 (P_{pid} - x_{i,d}(t)) + c_2 r_2 (P_{gd} - x_{i,d}(t)) \quad (1)$$

$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1) \quad (2)$$

where c_1 and c_2 are two learning factors which control the influence of the social and cognitive components and $r_i = \mathtt{rand}_i$, ($i = 1, 2$) are numbers independently generated within the range of [0,1]. $v_{i,d}(t)$ is the velocity of individual i on dimension d. $x_{i,d}(t)$ is current particle position on dimension d. P_{pid} (*pbest*) is the best local position of individual i on dimension d, and P_{gd} (*gbest*) represents the best particle position among all the particles in the population on dimension d. ω is the inertia weight, which ensures the convergence of the PSO algorithm.

According to Eq. (2), changes in the position of particles depend exclusively upon the position item of the PSO. Therefore, we only use the position item to investigate the search ability of particles during iterations. The implicit form of the position equation presented in Eq. (2) is used for a multi-particle PSO working in a multi-dimensional search space. Since the data of each dimension in the PSO are independent, the analysis below will be restricted to a single dimension. We simplify the Eqs. (1) and (2) as follow:

$$V(t+1) = \omega V(t) + c_1 r_1 (P_p - X(t)) + c_2 r_2 (P_g - X(t)) \quad (3)$$

$$X(t+1) = X(t) + V(t+1) \quad (4)$$

The following formulas can be obtained from Eq. (4):

$$V(t+1) = X(t+1) - X(t), \quad V(t) = X(t) - X(t-1) \quad (5)$$

By substituting Eq. (5) into Eq. (3), the following non-homogeneous recurrence relation is obtained:

$$X(t+1) = (1 + \omega - c_1 r_1 - c_2 r_2) X(t) - \omega X(t-1) + (c_1 r_1 P_p - c_2 r_2 X(t) P_g) \quad (6)$$

$$X(t+1) = (1+\omega) X(t) - \omega X(t-1) + (c_1 r_1 P_p + c_2 r_2 P_g) - (c_1 r_1 + c_2 r_2) X(t) \quad (7)$$

Let $\varepsilon = (c_1 r_1 P_p + c_2 r_2 P_g) - (c_1 r_1 + c_2 r_2) X(t)$, then

$$X(t+1) = (1+\omega) X(t) - \omega X(t-1) + \varepsilon \quad (8)$$