Applied Soft Computing 47 (2016) 534–552

Contents lists available at ScienceDirect

Applied Soft Computing

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

Multi swarm bare bones particle swarm optimization with distribution adaption

Reza Vafashoar*, Mohammad Reza Meybodi

Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 9 November 2015
Received in revised form 30 April 2016
Accepted 19 June 2016
Available online 23 June 2016

Keywords:
Particle swarm optimization
Multi-swarm PSO
Global numerical optimization
Learning automata
Cellular learning automata

ABSTRACT

Bare bones PSO is a simple swarm optimization approach that uses a probability distribution like Gaussian distribution in the position update rules. However, due to its nature, Bare bones PSO is highly prone to premature convergence and stagnation. The characteristics of the probability distribution functions used in the update rule have a tense impact on the performance of the bare bones PSO. As a result, this paper investigates the use of different methods for estimating the probability distributions used in the update rule. Four methods or strategies are developed that are using Gaussian or multivariate Gaussian distributions. The choice of an appropriate updating strategy for each particle greatly depends on the characteristics of the fitness landscape that surrounds the swarm. To deal with issue, the cellular learning automata model is incorporated with the proposed bare bones PSO, which is able to adaptively learn suitable updating strategies for the particles. Through the interactions among its elements and the learning capabilities of its learning automata, cellular learning automata gradually learns to select the best updating rules for the particles based on their surrounding fitness landscape. This paper also, investigates a new and simple method for adaptively refining the covariance matrices of multivariate Gaussian distributions used in the proposed updating strategies. The proposed method is compared with some other well-known particle swarm approaches. The results indicate the superiority of the proposed approach in terms of the accuracy of the achieved results and the speed in finding appropriate solutions.

1. Introduction

Particle swarm optimization (PSO) is a stochastic search approach inspired from the intelligent social behavior that exists among some animals like bird and fish [1]. It consists of a swarm of particles flying in a search space for better solutions. The flight which is guided by the velocity of particles resembles an iterative search process. Particles adjust their velocities in this iterative search based on their own and social experiences. They can memorize their historical best visited locations and the historical best visited location of the whole swarm.

Traditional PSO suffers from premature convergence and low exploration ability, especially, when dealing with complex multi modal problems [2,3]. As a result, many researches proposed various approaches to remedy this problem. Parameter adaption [4–7], hybridization [8–10], incorporation of different updating, selection, mutation strategies [11–13], topology variations [14–17], and integration of learning schemes [18–21] are some of the well-studied methods.

Studying the population dynamics and its convergence behavior [22], Kennedy suggested that the new positions of particles can be obtained by sampling from an explicit probability distribution rather than using velocity information. In their work, the Gaussian distribution is used for the acquisition of new positions, and its parameters for a particular particle are exclusively defined by the informers (particles utilized in its position updating) of the particle. The mean of the Gaussian distribution used in the position update rule of a particular particle is defined as the average of the particle's own historical best position and the swarm historical best position. Also, the distance of these two positions defines the standard deviation of the distribution [23].

Bare bones PSO (BBPSO) still suffers from slow and premature convergence and stagnation. In the latter case, the swarm converges to non-optimal points [24–27]. Considering the definition of updating distribution, the best particle of each iteration remains unchanged during that iteration, which may result in stagnation. BBPSO with mutation and crossover operations (BBPSO-MC) handles this issue by employing the mutation strategy of differential evolution algorithms in the update rule of the best particle [24].

* Corresponding author.
E-mail addresses: vafashoar@aut.ac.ir (R. Vafashoar), mmeybodi@aut.ac.ir (M.R. Meybodi).

http://dx.doi.org/10.1016/j.asoc.2016.06.028
1568-4946/

Other mutation strategies like Gaussian or Cauchy are also examined for improving bare bones PSO [25,28,29]. Krohling and Mendel proposed a jumping mechanism for avoiding local optima. Their jumping mechanism is based on the Gaussian or Cauchy noises which are added to the position of particles when they don't improve for a period of time [28].

Hybridizing bare bones PSO with other methods is also studied by some researchers. Gao et al. introduced a bee colony algorithm which produces candidate individuals in the onlooker phase based on the update strategies of bare bones PSO. Because the particles in bare bones PSO are attracted to the best individuals, the authors considered this step as an improved exploitation phase [30].

Chen introduced a variant of bare bones PSO which uses a mixture of two centers in the Gaussian distribution updating the particle positions. One of the centers is using local best positions and is considered to have exploration characteristics, while the other is based on the global best position and is used for better exploitation around the best particle. At early iterations, exploration has a bigger weight which gradually decreases in the favor of exploitation [31,32].

Some other variants of bare bones PSO utilize heavy tail probability distributions instead of Gaussian distribution for more explorative behavior [25,29]. Li and Yao introduced a Cooperative Particle Swarms optimization method (CCPSO) for Large Scale Optimization. The population of CCPSO is divided into several swarms, and the swarms are updated cooperatively. The position of a particle at each step is updated similar to bare bones PSO; however, it uses different probability distributions for position sampling: a Gaussian distribution with its center at the best historical position of the particle itself, and a Cauchy distribution whose center is at the local best position. Either of the two distributions will be selected with a fixed probability for updating the particle position in each dimension. CCPSO, also, changes the number of swarms when the global best fitness value is not improving; this way, it can adaptively adjust the number of swarms. BBPSO with scale matrix adaptation (SMA-BBPSO) uses multivariate t-distribution with adaptive scale matrices in its update rule. It employs a simple, yet effective, method for adapting the scale matrices associated with the particles: the best position found by any particle in the neighborhood of a particular particle is sampled in its associated scale matrix for the next iteration.

This paper investigates the use of various probability models for updating the positions of particles in BBPSO. Each model can benefit the search process depending on the shape of the landscape where the particles reside. Some of the studied models involve multivariate Gaussian distributions for updating particle positions. Accordingly, a simple method for adapting the covariance matrices of these Gaussian distributions will be introduced. To achieve a good performance, a strategy selection mechanism is needed. This mechanism should select an appropriate updating strategy for each particle based on the properties of the particle's surrounding landscape. To achieve this objective, the swarms are embedded in a cellular learning automata (CLA) [33–36], which adaptively learns the best updating models for each particle. CLA is a hybrid mathematical model for many decentralized problems (decentralized problems can be considered as multiple systems where each system has its own processing unit, and can make decisions according to on its own knowledge). It can be considered as a network of agents with learning abilities. These agents have the ability to interact and learn from each other through a set of CLA rules. CLA has been utilized in many scientific domains such as deployment and clustering in wireless sensor networks [37,38], dynamic channel assignment in cellular networks [39], evolutionary computing [40], and graph coloring [41].

The rest of the paper is organized as follows: Section 2 gives a brief review on the involved concepts such as learning automata, cellular learning automata and particle swarm algorithm. Section 3 gives the detailed description of the proposed approach. Experimental results and comparison to other algorithms comes in Section 4, and finally Section 5 contains conclusions.

2. The involved concepts

This section provides a brief description of the concepts that are used in the paper. It starts with an introduction on particle swarm optimization and Bare Bones PSO; then, reviews the basic concepts of cellular learning automata.

2.1. PSO and related topics

Traditional PSO utilizes a population of particles called swarm. Each particle X_i in the swarm keeps information about its current position, its personal best visited position known as pbest, and its flying velocity, respectively, in three D dimensional real valued vectors: $x_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]$, $p_i = [p_{i1}, p_{i2}, \ldots, p_{iD}]$, and $v_i = [v_{i1}, v_{i2}, \ldots, v_{iD}]$. All particles know the best position experienced by the whole swarm known as gbest, which is also represented by a vector like $p_g = [p_{g1}, p_{g2}, \ldots, p_{gD}]$. The search in a D dimensional space is an iterative process, during which the velocity and position of a particle like X_i are updated according to the following equations:

$$\begin{aligned} v_{id}(k+1) &= \omega v_{id}(k) + c_1 r_1 (p_{id} - x_{id}(k)) + c_2 r_2 (p_{gd} - x_{id}(k)) \\ x_{id}(k+1) &= x_{id}(k) + v_{id}(k+1) \end{aligned} \quad (1)$$

where $v_{id}(k)$ is the dth dimension of the velocity vector of the particle in step k; $x_{id}(k)$ and $p_{id}(k)$ are respectively the dth dimension of its position and historical best position vectors; $p_{gd}(k)$ represents the dth dimension of the historical best position of the whole swarm in step k; ω is the inertia weight, which was introduced to bring a balance between the exploration and exploitation characteristics [3]; c_1 and c_2 are acceleration constants that represent cognitive and social learning weights; and, finally, r_1 and r_2 are two random numbers from the uniform distribution $u(0, 1)$. After acquiring the new positions of the particles, the historical best position of each particle is updated; which may, also, affect the historical global best position of the swarm.

2.2. Bare bones particle swarm optimization

The observation of the motion of individual particles in the original PSO inspired the development of bare bones PSO. In traditional PSO, the distribution of the positions visited by a single particle resembles the bell shaped curve of a Gaussian distribution. Accordingly, BBPSO uses Gaussian probability distribution to generate the new position of a particle like X_i [23]:

$$x_{id}(k+1) = \frac{p_{id} + pl_{id}}{2} + N(0,1) \times |p_{id} - pl_{id}| \quad (2)$$

where $N(0,1)$ denotes a random number taken from the normal distribution with mean 0 and standard deviation 1. pl is the best location, visited so far, in the neighborhood of the particle X_i, and p_i is the personal best location of the particle itself.

2.3. Learning automaton

A variable structure learning automaton (LA) can be represented by a sextuple like $\{\Phi, \alpha, \beta, A, G, P\}$ [42]. where Φ is a set of internal states; α is a set of outputs or actions of the learning automaton; β is a set of inputs or environmental responses; A is a learning algorithm; $G(.)$: $\Phi \rightarrow \alpha$ is a function that maps the current state into the current output; and P is a probability vector that determines the selection probability of an state at each stage. There is usually a one