Expert Systems with Applications 41 (2014) 6459–6466

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Review

A survey on nature-inspired optimization algorithms with fuzzy logic for dynamic parameter adaptation

Fevrier Valdez, Patricia Melin *, Oscar Castillo

Tijuana Institute of Technology, Tijuana, Mexico

ARTICLE INFO

Keywords:
Particle Swarm Optimization
Gravitational Search Algorithm
Ant Colony Optimization
Fuzzy logic

ABSTRACT

Metaheuristic optimization algorithms have become a popular choice for solving complex problems which are otherwise difficult to solve by traditional methods. However, these methods have the problem of the parameter adaptation and many researchers have proposed modifications using fuzzy logic to solve this problem and obtain better results than the original methods. In this study a comprehensive review is made of the optimization techniques in which fuzzy logic is used to dynamically adapt some important parameters in these methods. In this paper, the survey mainly covers the optimization methods of Particle Swarm Optimization (PSO), Gravitational Search Algorithm (GSA), and Ant Colony Optimization (ACO), which in the last years have been used with fuzzy logic to improve the performance of the optimization methods.

1. Introduction

Optimization is the process of making something better. An engineer or scientist proposes up a new idea and optimization improves on that idea. Optimization consists in trying variations on an initial concept and using the information gained to improve on the idea. Optimization problems can be seen in many areas such as manufacturing systems, economics, physical sciences, computational systems, etc.

In the last years, several optimization methods have been developed based on the nature inspired analogy. However, these methods are not always able to solve some problems in the best way. Although it has been shown that these are good methods to solve complex problems, there are not yet methods to know the optimal parameters to solve problems that can be set at the beginning when using the algorithms. As an attempt to improve these methods many researchers have used fuzzy logic to adapt parameters and achieve better results than with the original methods. Therefore in this review we are including the optimization methods with parameter adaptation using fuzzy logic. Several works have been developed in the last years using fuzzy systems for parameter adaptation, for example in Castillo, Valdez, and Melin (2007) a hierarchical genetic algorithm for topology optimization in fuzzy control systems is used. In this survey we only cover the PSO, GSA an ACO nature-inspired optimization methods of and our paper is focused on the algorithms that use fuzzy systems to improve the parameter adaptation on the optimization methods. However, a chronological order of the most popular nature-inspired optimization methods is shown in Table 1. In Section 2 a brief description about of the nature-inspired optimization methods is presented, in this case PSO, GSA and ACO. In Section 3, the main topic of this paper is described, which is a survey on nature-inspired optimization methods with fuzzy parameter adaptation. In Section 4, a summary of the papers analyzed is presented. In Section 5 the conclusions are shown. The choice of these 3 optimization methods was made because there are several works in which it has been demonstrated that the extension of these metaheuristics with fuzzy systems is a good alternative to improve the original methods. Therefore we included in this survey the papers more recently found in the literature on this subject.

2. A brief description of the nature-inspired optimization methods

In this section we briefly describe the methods discussed in this paper. In this case, the methods of study, which are Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and the Gravitational Search Algorithm (GSA). The following subsections briefly describe the basic theory of each method in its original form. This description is considered necessary to grasp the ideas behind the use of fuzzy logic in enhancing the original metaheuristic methods by providing them with dynamic parameter adaptation capabilities.

* Corresponding author. Tel.: +52 6646236318.
E-mail addresses: fevrier@tectijuana.mx (F. Valdez), pmelin@tectijuana.mx (P. Melin), ocastillo@tectijuana.mx (O. Castillo).

http://dx.doi.org/10.1016/j.eswa.2014.04.015
0957-4174/

Table 1
Popular optimization methods in chronological order.

Year	Nature-Inspired optimization methods
2012	Krill Herd (Gandomi & Alavi, 2012; Gandomi et al., 2013; Wang, Gandomi, & Alavi, 2014; Guo, Wang, Gandomi, Alavi, & Duan, 2014)
2010	Bat Algorithm (Yang, 2010; Yang & Gandomi, 2012; Gandomi & Yang, 2014; Gandomi et al., 2013)
2010	Artificial Bee Algorithm (Karaboga & Akay, 2009)
2009	Cuckoo Search Algorithm (Yang, 2014; Bulatović, Đorđević, & Đorđević, 2013; Gandomi et al., 2013)
2009	Gravitational Search Algorithm (Rashedi & Nezamabadi-pour, 2009)
2007	Firefly Algorithm (Gandomi & Alavi, 2011; Yang, Soheil, Hosseini, & Gandomi, 2012; Gandomi, Yang, Talatahari, & Alavi, 2013; Talatahari, Gandomi, & Yun, 2014)
2007	Intelligent water drops (Shah-Hosseini, 2009)
2005	Harmony Search Algorithm (Geem, 2008)
2005	Honey Bee Algorithm (Teodorovic & Dell'orco, 2005)
2002	Bacterial Foraging Algorithm (Tang, Wu, & Saunders, 2006)
2002	Estimation of Distribution Algorithm (Ocenasek & Schwarz, 2002)
1995	Particle Swarm Optimization (Eberhart & Kennedy, 1995)
1992	Ant Colony Optimization (Colorni, Dorigo, & Maniezzo, 1992)
1989	Tabu Search Glover, 1989
1983	Simulated Annealing (Kirkpatrick, Gelatt, & Vecchi, 1983)
1979	Cultural Algorithms (Reynolds, 1994)
1975	Genetic Algorithms (Holland, 1975)
1966	Evolutionary Programming (Melin et al., 2013)
1965	Evolution Strategies (Kennedy et al., 1995)

2.1. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population based stochastic optimization technique developed by Kennedy and Eberhart in (1995), inspired by the social behavior of bird flocking or fish schooling (Kennedy & Eberhart, 1995).

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA) Castillo & Melin, 2002. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike the GA, the PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles (Angeline, 1998).

Each particle keeps track of its coordinates in the problem space, which are associated with 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 neighbors of the particle. This location is called *lbest*. When a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest*.

The Particle Swarm Optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its p*best* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations.

In the past several years, PSO has been successfully applied in many research and application areas. It has been experimentally shown that PSO gets better results in a faster, cheaper way compared with other methods (Angeline, 1998).

Another reason that PSO is attractive is that there are only few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle Swarm Optimization has been considered for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

The pseudo code of the PSO is as follows

```
For each particle
  Initialize particle
End
Do
  For each particle
    Calculate fitness value
    If the fitness value is better than the best fitness value (pBest)
  in history
      set current value as the new pBest
  End
  Choose the particle with the best fitness value of all the
  particles as the gBest
  For each particle
    Calculate particle velocity
    Update particle position
  End
While maximum iterations or minimum error criteria is not
  attained
```

2.2. Gravitational Search Algorithm

The original gravitational approach was proposed for Rashedi et al., where they introduce a new algorithm based on populations and at the same time it takes as fundamental principles the law of gravity and second motion law. Its principal features are that agents are considered as objects and their performance is measured by their masses, all these objects are attract each other by the gravity force, and this force causes a global movement of all objects, the masses cooperate using a direct form of communication, through gravitational force, an agent with heavy mass correspond to good solution therefore its move more slowly than lighter ones, finally its gravitational and inertial masses are determined using a fitness function. This algorithm considers each mass as a solution and the algorithm is navigated by properly adjusting the gravitational and inertia masses, throughout iterations expect that masses be attracted by the heaviest mass. This mass will present an optimum solution in the search space (Ahmadizar & Soltanpanah, 2011).

The way in which the position of number N of agents is represented by

$$X_i = (x_i^1, \ldots, x_i^d, \ldots, x_i^n) \quad \text{for } i = 1, 2, \ldots, N, \tag{1}$$

where x_i^d presents the position of i_{th} agent in the d_{th} dimension.

Now, Eq. (1) with the new concepts of masses is defined as following: the force acting on mass i from mass j in a specific time t, is given by

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon}(x_j^d(t) - x_i^d(t)) \tag{2}$$

where M_{aj} is the active gravitational mass related to agent j, M_{pi} is the passive gravitational mass related to agent i, $G(t)$ is gravitational constant at time t, ε is a small constant, and $R_{ij}(t)$ is the Euclidean distance between two agents i and j:

$$R_{ij}(t) = \|x_t(t), x_j(t)\|_2 \tag{3}$$

The stochastic characteristic of this algorithm is based on the idea of the total force that acts on agent i in a dimension d be a randomly weighted sum of dth components of the forces exerted from other agents,

$$F_i^d(t) = \sum_{j=i,\, j\neq i}^{N} rand_j F_{ij}^d(t) \tag{4}$$