Knowledge-Based Systems 82 (2015) 60–69

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

An improved simplified swarm optimization

Wei-Chang Yeh

Integration and Collaboration Laboratory, Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Taiwan

ARTICLE INFO

Article history:
Received 3 February 2014
Received in revised form 21 January 2015
Accepted 22 February 2015
Available online 4 March 2015

Keywords:
Swarm Intelligence (SI)
Simplified Swarm Optimization (SSO)
Artificial Bee Colony Algorithm (ABC)
Soft Computing (SC)
Optimization problems

ABSTRACT

This paper introduces an improved simplified swarm optimization (iSSO) by undertaking a major revision of the update mechanism (UM) of traditional SSO. To test its performance, the proposed iSSO is compared with another recently introduced swarm-based algorithm, the Artificial Bee Colony Algorithm (ABC), on 50 different widely used multivariable and multimodal numerical test functions. Numerical examples conclude that the proposed iSSO outperforms ABC in both solution quality and efficiency. We also test the roles of the proposed UMs and iterative local search. The proposed algorithm is thus useful to both practitioners and researchers.

1. Introduction

Since the early 1990s, Soft Computing (SC) has been utilized to obtain optimal or good-quality solutions to difficult optimization problems in fields for which exact analytical methods do not produce optimal solutions in an acceptable amount of time, especially for problems that are large in size [51]. It is well known that SC mainly involves fuzzy sets, neural networks, genetic algorithms, and rough sets. In recent years, we have seen an increasing interest in SC in creating methodologies with the aims of resembling and simulating phenomena of nature to solve larger problems in science and technology; resulting methods include Artificial Neural Network (ANN) [9,25], Simulated Annealing (SA) [23], Tabu Search (TS) [13], Genetic Algorithm (GA) [10,14], Ant Colony Optimization (ACO) [7], Particle Swarm Optimization (PSO) [21,22,28,12,40], Estimation Distribution of Algorithm (EDA) [24], Differential Evolution [31,32], Artificial Bee Colony Algorithm (ABC) [16,17,19,20], Simplified Swarm Optimization (SSO) [44–50], Firefly algorithm [41], Biogeography-based Optimization [29], Cuckoo search [42,39], Bat Algorithm [43], and Krill Herd [11,15,34–38].

Swarm Intelligence (SI) is a newly developed branch of SC that belongs to the category of population-based stochastic optimization. Swarm Intelligence (SI) was introduced by Gerardo Beni and Jing Wang in 1989 in the context of cellular robotic systems [3]. Bonabeau, Dorigo and Theraulaz defined SI as "any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies" [4]. SI has also received significant interest from researchers studying in a range of research areas and has been applied to several real-world problems.

ABC is one of the most recently introduced SI algorithms and was originally designed by Karabog in 2005 [19] to simulate the intelligent foraging behavior of a honeybee swarm, based on the model proposed by Tereshko and Loengarov [33]. Research in the literature indicates that the performance of the ABC is better than or similar to that of other population-based algorithms, including GA, PSO, and EDA [16,17,19,20]. Hence, ABC is selected to compare with the proposed approach.

SSO is an emerging population-based stochastic optimization method [44–48,2,1,6,5,27,30,26,8]. It is another new SI method and is also an evolutionary computation method. SSO was proposed by Yeh to overcome the drawbacks of PSO [44]. The advantages of SSO are its simplicity (easy to implement and having only a few parameters to tune), efficiency (a fast convergence rate), and flexibility. Simulation results reveal that SSO has better convergence to quality solutions than PSO, GA, EDA, and ANN [44–48]. SSO has therefore attracted considerable attention and has been widely used in a range of applications, such as redundancy allocation [44,30], data mining [45,48,27,26], health care management [6,5,8], and supply chain management [46,47,2,1].

The goal of this work is to propose an improved SSO (here termed iSSO). iSSO is able to address non-discrete data. To demonstrate the efficiency of the proposed iSSO, a comprehensive comparative study is presented on the performances of the well-known swarm-based algorithms SSO and ABC and the variant

E-mail address: yeh@ieee.org

http://dx.doi.org/10.1016/j.knosys.2015.02.022
0950-7051/

iSSO for optimizing a large set of multi-variable and multi-modal numerical test functions. For iSSO, there is also consideration of using iterative local search (ILS) to the best solution, ILS to all improved solutions, or neither.

This paper is organized as follows. Section 2 provides respective descriptions of ABC, which may be the best-known SC for solving numerical functions, and SSO, which is the basis of the proposed iSSO and related UM. The proposed UM and iSSO are discussed in Section 3. A comprehensive comparative study on the performances of the proposed UM with or without ILS and SSO based on 50 benchmark functions is given in Section 4. The complete comparison of the proposed iSSO and ABC, together with the proposed UM, is discussed in Section 5. Finally, the conclusion and discussion are given in Section 6.

2. Overview of ABC and SSO

Let Nvar, Nsol, Ngen, and Nrun represent the number of variables (attributes, dimensions, genes), solutions, (populations chromosomes, particles), generations (called cycles in ABC), and independent replications. Most SCs, e.g., GA, SA, ACO, PSO, ABC, SSO, and the proposed iSSO, generate randomly distributed initial solutions with population size Nsol inside the problem space and then search for optimal solutions by updating generations. Each solution is encoded as a finite-length string with a fitness value. Note that large Nsol and Ngen will slow down the optimization process significantly, and small Nsol and Ngen will be unable to find a good solution for the requirements.

Let $X_i^t = (x_{i1}^t, x_{i2}^t, \ldots, x_{i,\mathrm{Nvar}}^t)$ be the ith solution (called a food source in ABC) at generation t, x_{ij}^t be the jth variable of X_i^t, and $f(X_i^t)$ be the fitness function of X_i^t. The major difference among all SC methods is the UM that is used to update the current solutions. Hence, we focus on the UMs of ABC, SSO, and iSSO in the rest of the study. Sometimes, ILSs are implemented after using UMs to improve solutions by moving them to their neighbors according to neighborhood structures, but this is done at the expense of running time.

2.1. ABC

In ABC, the quality of the possible solution X_i^t is denoted by $F(X_i^t)$ and defined as follows for $i = 1, 2, \ldots, \mathrm{Nsol}$ and $t = 1, 2, \ldots, \mathrm{Ngen}$ [16,17,19,20].

$$F(X_i^t) = \begin{cases} \frac{1}{f(X_i^t)+1}, & f(X_i^t) \geqslant 0 \\ 1 + |f(X_i^t)|, & \text{otherwise} \end{cases} \quad (1)$$

There are three UMs, called the employed bee, the onlooker (bee), and the scout (bee), in ABC. The first half of the bee colony implements employed bees, while the second half implements onlookers. ABC assumes only one employed bee per food source.

In the first UM, "the employed bees," X_i^t is updated to X_i^{t+1} based on the following equation:

$$x_{ik}^{t+1} = x_{ik}^t + \rho_{[-1,1]} \cdot (x_{ik}^t - x_{jk}^t), \quad (2)$$

where both the variable k and solution X_j^t are selected randomly and ρ_I is a uniform variable in the interval I, where $i, j = 1, 2, \ldots, \mathrm{Nsol}$, $k = 1, 2, \ldots, \mathrm{Nvar}$, and $t = 0, 1, 2, \ldots, \mathrm{Ngen} - 1$. If x_{ik}^{t+1} falls outside the acceptable range, it is set to the corresponding extreme value in that range. Moreover, if $F(X_i^{t+1})$ is worse than $F(X_i^t)$, then X_i^{t+1} is replaced by X_i^t (called a greedy selection mechanism in ABC).

To guide ABC toward more highly fit positions of the search space, the second UM, "the onlooker," is a local search operator. After a solution, say, X_i^t, is generated or updated, the onlooker bee is implemented Nsol times with probability $Pr(X_i^t)$ defined as follows:

$$Pr(X_i^t) = \frac{F(X_i^t)}{\sum_{i=1}^{\mathrm{Nsol}} F(X_i^t)}. \quad (3)$$

If a solution, say, X_i^t, does not improve for a predetermined number of iterations

$$\mathrm{Nlim} = \mathrm{Nsol} \times \mathrm{Nvar}, \quad (4)$$

then the third UM, "the scout," is a special ILS that is implemented to regenerate X_i^t randomly and repeatedly.

2.2. SSO

The proposed iSSO is based on SSO. Before discussing the proposed iSSO, basic SSO is introduced formally in this sub-section. Let *pBest* $P_i = (p_{i1}, p_{i2}, \ldots, p_{i,Nvar})$ be the best fitness function value of the ith solution with its own history and *gbest* $G = (g_1, g_2, \ldots, g_{Nvar})$ be the solution with the best fitness function value among all *pBest*s, where $i = 1, 2, \ldots, \mathrm{Nsol}$.

The fundamental concept of SSO is that each variable of any solution needs to be updated to a value related to its current value, its current *pBest* (as a local search), the *gBest* (as a global search), or a random feasible value to maintain population diversity and enhance the capacity to escape from a local optimum [44–48]. The UM of SSO is based only on the following simple mathematical modeling after c_w, c_p, and c_g are given:

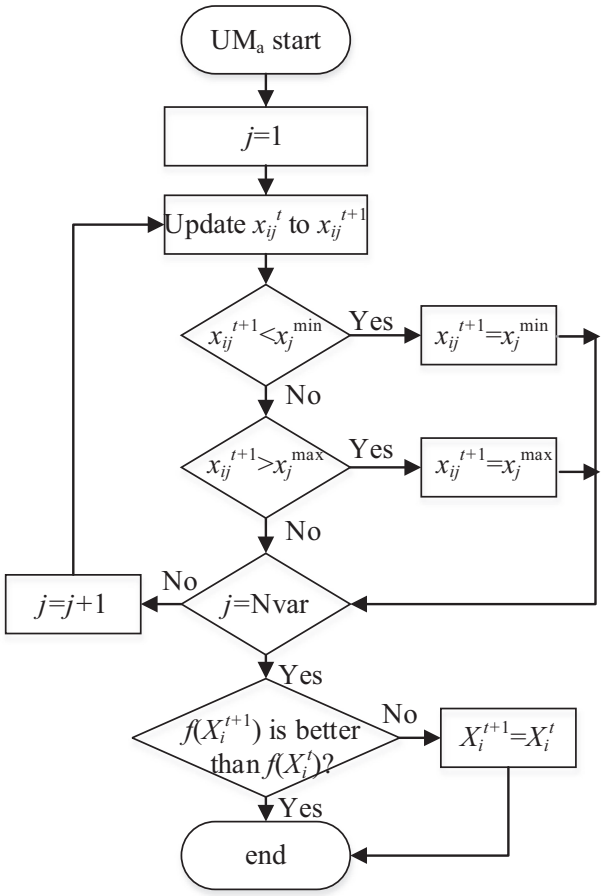

Fig. 1. The flowchart of the proposed $\mathrm{UM_a}$.