Knowledge-Based Systems 62 (2014) 69–83

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

An improved fruit fly optimization algorithm for continuous function optimization problems

Quan-Ke Pan [a,b], Hong-Yan Sang [b], Jun-Hua Duan [b], Liang Gao [c,*]

[a] State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, PR China
[b] College of Computer Science, Liaocheng University, Liaocheng 252059, PR China
[c] State Key Laboratory of Digital Manufacturing Equipment & Technology, Huazhong University of Science & Technology, Wuhan 430074, PR China

ARTICLE INFO

Article history:
Received 4 September 2013
Received in revised form 6 February 2014
Accepted 26 February 2014
Available online 12 March 2014

Keywords:
Fruit fly optimization
Evolutionary algorithms
Meta-heuristics
Continuous optimization
Harmony search

ABSTRACT

This paper presents an improved fruit fly optimization (IFFO) algorithm for solving continuous function optimization problems. In the proposed IFFO, a new control parameter is introduced to tune the search scope around its swarm location adaptively. A new solution generating method is developed to enhance accuracy and convergence rate of the algorithm. Extensive computational experiments and comparisons are carried out based on a set of 29 benchmark functions from the literature. The computational results show that the proposed IFFO not only significantly improves the basic fruit fly optimization algorithm but also performs much better than five state-of-the-art harmony search algorithms.

1. Introduction

Fruit fly optimization (FFO) is one of the latest meta-heuristic methods presented in the literature. FFO simulates the intelligent foraging behavior of fruit flies or vinegar flies in finding food. It was proposed by Pan [1,2] for global optimization. Fruit flies live in the temperate and tropical climate zones. They have very sensitive osphresis and vision organs which are superior to other species. They feed chiefly on rotten fruits. In the process of finding food, their osphresis organs smell all kinds of scents in the air. They then fly towards the corresponding locations. When they get close to the food locations, they find foods using their visions and then fly towards that direction.

The FFO algorithm has many advantages such as a simple structure, immediately accessible for practical applications, ease of implementation and speed to acquire solutions. Therefore, it has captured much attention and has been successfully applied to solve a wide range of practical optimization problems including the financial distress [2], power load forecasting [3], web auction logistics service [4], PID controller tuning [5,6], and multidimensional knapsack problem [7]. However, to the best of our knowledge, the FFO algorithm has not yet been used to solve high-dimensional continuous function optimization problems which are used extensively in science, engineering, and finance. Most of these problems are no longer linear, quadratic, nor unimodal. Their objective functions are often multimodal with peaks, valleys, channels, and flat hyper-planes of different heights. Solving these types of problems to optimality undoubtedly becomes a true challenge [8].

In this paper, we adapt the FFO algorithm to high-dimensional functions and present an improved variant, namely improved fruit fly optimization (IFFO). Some advanced techniques are introduced to improve the effectiveness of the FFO algorithm including a new control parameter and an effective solution generating method. The IFFO is applied to various standard optimization problems and compared with the basic FFO, a variant of the FFO, and five existing harmony search (HS) algorithms. Numerical results reveal that the proposed IFFO is a new powerful search algorithm for various optimization problems.

The rest of the paper is organized as follows. In Section 2, the basic FFO algorithm is introduced, followed by the presented IFFO in Section 3. Section 4 lists a total of 29 benchmark functions. Experimental design and comparisons are presented in Section 5. Finally, Section 6 gives the concluding remarks.

2. The basic fruit fly optimization algorithm

The basic FFO consists of four consecutive phases. These are initialization, osphresis foraging, population evaluation, and vision

* Corresponding author. Tel.: +86 27 87544522.
E-mail address: gaoliang@mail.hust.edu.cn (L. Gao).

http://dx.doi.org/10.1016/j.knosys.2014.02.021
0950-7051/

foraging. Firstly, FFO sets its control parameters, i.e., population size and a termination criterion, and initializes its fruit fly swarm location. Then FFO is repeated with the search process of osphresis foraging and vision foraging until the termination criterion is satisfied. At the osphresis foraging phase, a population of fruit flies randomly search food sources around the fruit fly swarm location. After that, the smell concentration value or fitness value is evaluated for each of the food sources. In the vision foraging phase, the best food source with the maximum smell concentration value is found and then the fruit fly group flies towards it. The basic FFO was presented for the financial distress [2]. We adapt it to solve high-dimensional function optimization problems as follows.

2.1. Initialize the problem and algorithm parameters

In general, the global optimization problem can be summarized as follows.

$\min\ f(X)$

$$\text{st}: \ x_j \in [LB_j, UB_j], \quad j = 1, 2, \ldots, n \tag{1}$$

where $f(X)$ is the objective function, $X = (x_1, x_2, \ldots, x_n)$ is the set of decision variables, n is the number of decision variables, and LB_j and UB_j are the lower and upper bounds for the decision variable x_j, respectively.

The parameters of the FFO algorithm are the population size (PS) and the maximum number of iterations ($Iter_{\max}$). It is obvious that a good set of parameters can enhance the algorithm's ability to search for the global optimum or near optimum region with a high convergence rate.

2.2. Initialize the fruit fly swarm location

The fruit fly swarm location, $\Delta = (\delta_1, \delta_2, \ldots, \delta_n)$, is randomly initialized in the search space as follows.

$$\delta_j = LB_j + (UB_j - LB_j) \times rand(), \quad j = 1, 2, \ldots, n \tag{2}$$

where $rand()$ is a function which returns a value from the uniform distribution on the interval [0,1].

2.3. Osphresis foraging phase

In the osphresis foraging phase, a population of PS food sources are generated randomly around the current fruit fly swarm location Δ. Let $\{X_1, X_2, \ldots, X_{PS}\}$ represent the generated food sources, where $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n})$, $i = 1, 2, \ldots, PS$, is yielded as follows.

$$x_{i,j} = \delta_j \pm rand(), \quad j = 1, 2, \ldots, n \tag{3}$$

2.4. Vision foraging phase

In the vision foraging phase, FFO carries out a greedy selection procedure. The best food source with the lowest fitness, X_{best}, is first found, i.e., $X_{best} = \arg(\min_{i=1,2,\ldots,PS} f(X_i))$. If X_{best} is better than the current fruit fly swarm location Δ, it will replace the swarm location and become a new one in the next iteration, i.e., $\Delta = X_{best}$ if $f(X_{best}) < f(\Delta)$.

2.5. Check stopping criterion

If the stopping criterion (maximum number of iterations) is satisfied, computation is terminated. Otherwise, the osphresis foraging and vision foraging phases are repeated.

2.6. Procedure of the basic FFO

The complete computational procedure of the presented FFO algorithm is outlined in Fig. 1. Note in the algorithm, $\Delta = (\delta_1, \delta_2, \ldots, \delta_n)$ is the fruit fly swam location.

Algorithm 1. The FFO algorithm
Parameters: Population size (PS) and the maximum iterations ($Iter_{\max}$)
Output: Solution X^*

```
//Initialization
Set PS and Iter_max
//Initialize swarm location Δ = (δ_1, δ_2, .., δ_n)
δ_j = LB_j + (UB_j − LB_j) × rand( ),  j = 1, 2, ..., n
Iter = 0
X* = Δ
Repeat
  //Osphresis foraging phase
  For i = 1, 2, ..., PS
    //Generate food source X_i = (x_{i,1}, x_{i,2}, ..., x_{i,n})
    x_{i,j} = δ_j ± rand( ),  j = 1, 2, ..., n
    If x_{i,j} > UB_j then x_{i,j} = UB_j
    If x_{i,j} < LB_j then x_{i,j} = LB_j
  Endfor

  //Vision foraging phase
  X_best = arg( min_{i=1,2,...,PS} f(X_i) )
  If f(X_best) < f(Δ) then Δ = X_best
  If f(Δ) < f(X*) then X* = Δ
  Iter = Iter + 1
Until Iter == Iter_max
```

Fig. 1. The basic FFO algorithm.