International Journal of Swarm Intelligence and Evolutionary Computation

Allah, Int J Swarm Intel Evol Comput 2016, 5:2
DOI: 10.4172/2090-4908.1000134

Research Article Open Access

Hybridization of Fruit Fly Optimization Algorithm and Firefly Algorithm for Solving Nonlinear Programming Problems

Rizk M Rizk Allah*

Department of Basic Engineering Science, El-Menoufia University, Shebin El-Kom, Egypt

Abstract

We propose a novel hybrsid algorithm named, FOA-FA to solve the nonlinear programming problems (NLPPs). The main feature of the hybrid algorithm is to integrate the strength of fruit fly optimization algorithm (FOA) in handling continuous optimization and the merit of firefly algorithm (FA) in achieving robust exploration. The methodology of the proposed algorithm consists of two phases. The first one employs a variation on original FOA employing a new adaptive radius mechanism (ARM) for exploring the whole scope around the fruit flies locations to overcome the drawbacks of original FOA which has been continues for the nonnegative orthant problems. The second one incorporates FA to update the previous best locations of fruit flies to avoid the premature convergence. The hybrid algorithm speeds up the convergence and improves the algorithm's performance. The proposed FOA-FA algorithm is tested on several benchmark problems and two engineering applications. The numerical comparisons have demonstrated its effectiveness and efficiency.

Keywords: Firefly algorithm; Fruit fly optimization algorithm; Nonlinear programming problems

Introduction

Traditional optimization methods can be classified into two distinct groups; direct and gradient-based methods. In direct search methods, only objective function and constraint value are used to guide the search, whereas gradient-based methods use the first and/or second-order derivatives of the objective function and/or constraints to guide the search process. Since derivative information is not used, the direct search methods are usually slow, requiring many function evaluations for convergence. For the same reason, they can be applied to many problems without a major change of the algorithm. On the other hand, gradient based methods quickly converge to an optimal solution, but are not efficient in non-differentiable or discontinuous problems. In addition, if there is more than one local optimum in the problem, the result may depend on the selection of an initial point, and the obtained optimal solution may not necessarily be the global optimum. Furthermore, the gradient search may become difficult and unstable when the objective function and constraints have multiple or sharp peaks.

Therefore, to overcome these shortcomings, a lot of research has focused on meta-heuristic methods. The meta-heuristic methods have many advantages compared to the traditional nonlinear programming techniques, among which the following three are the most important: (i) They can be applied to problems that consist of discontinuous, non-differentiable and non-convex objective functions and/or constraints. (ii) They do not require the computation of the gradients of the cost function and the constraints. (iii) They can easily escape from local optima. Many meta-heuristic algorithms such as, genetic algorithm (GA) [1], particle swarm optimization (PSO) [2], differential evolution [3] and ant colony optimization [4] have shown their efficacy in solving computationally intensive problems.

Among the existing meta-heuristic algorithms, a well-known branch is the FA which is a meta-heuristic search algorithm, has been developed by Yang [5,6]. FA mimics some characteristics of tropic firefly swarms and their flashing behavior. A firefly tends to be attracted towards other fireflies with higher flash intensity. The main advantages of FA are namely intensification and diversification or exploitation and exploration: As light intensity decreases with distance, the attraction among fireflies can be local or global, depending on the absorbing coefficient, and thus all local modes as well as global modes will be visited. Therefore, it has captured much attention and has been successfully applied to solve several optimization problems including [7-9].

Fruit fly optimization algorithm (FOA) is one of the latest meta-heuristic methods which proposed by Pan [10] for solving optimization problems. The main inspiration of FOA is that the fruit fly itself is superior to other species in sensing and perception, especially in osphresis and vision. 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. FOA has many advantages such as a simple structure, ease of implementation and speed to acquire solutions. Therefore, FOA has been successfully applied to solve several optimization problems [10-14].

Recently, hybridization is recognized to be essential aspect of high performing algorithms. Pure algorithms cannot reach to an optimal solution in a reasonable time. In addition, the premature convergence in pure algorithms is inevitable. Thus pure algorithms are almost always inferior to hybridizations. To our best knowledge, this is the first time that the hybridization between FOA and FA has been proposed for solving NLPPs. This hybridization aims to improve the performance of the original FOA and eliminate its drawbacks.

In this paper, we propose a novel hybrid algorithm named, FOA-FA for solving the nonlinear programming problems and two

***Corresponding author:** Rizk M Rizk Allah, Department of Basic Engineering Science, El-Menoufia University, Shebin El-Kom, Egypt, E-mail: rizk_masoud @sh-eng.menofia.edu.eg

Received March 21, 2016; **Accepted** April 29, 2016; **Published** April 30, 2016

Citation: Allah RMR (2016) Hybridization of Fruit Fly Optimization Algorithm and Firefly Algorithm for Solving Nonlinear Programming Problems. Int J Swarm Intel Evol Comput 5: 134. doi: 10.4172/2090-4908.1000134

engineering applications. The proposed FOA-FA algorithm employed the merits of both fruit fly optimization algorithm (FOA) and firefly algorithm (FA) and its methodology consists of two phases. The first one employs a variation on original FOA employing a new adaptive radius mechanism for exploring the whole scope around the fruit flies locations to overcome the drawbacks of original FOA which has been continues for the nonnegative orthant problems. In addition, several other improvements for original FOA are considered, such as: Initialization of the candidate solutions through the search space uniformly and a new modulation coefficient. Moreover, the premature convergence of original FOA degrades its performance by reducing its search capability, leading to a higher probability of being trapped to a local optimum. Therefore, the second phase incorporates the FA to update the previous best locations of fruit flies to force FOA jump out of premature convergence, because of its strong searching ability. Consequently, the hybrid algorithm speeds up the convergence and improves the algorithm's performance.

The rest of the paper is organized as follows. In Section 2 we describe some preliminaries of the nonlinear programming problem. In Section 3, we introduce briefly the basics of both FOA and FA algorithms. The hybrid algorithm named, FOA-FA algorithm is proposed and explained in details in Section 4. The numerical experiments are given in Section 5 to substantiate the superiority of the proposed algorithm. Finally we summarize the paper with some comments.

Nonlinear Programming Problem (NLPP)

The nonlinear programming problem can be defined as follow [15]:

$$NLPP: \quad \min_{\Omega} F(\mathbf{x}) \ \mathbf{x} = (x_1, x_2, ..., x_n) \in \Re^n.$$
$$\Omega = \left\{ \begin{array}{l} \mathbf{x} \mid g_k(\mathbf{x}) \le 0;\ h_j(\mathbf{x}) = 0; \\ k = 1,2,...,L;\ \ j = L+1,...,P \end{array} \right\} \quad (1)$$

Where $F(x)$ is the objective function defined on the search space S, $S \subseteq \Re^n$, X is called a decision variable and $g_k(x)$, $h_j(x)$ are defined the inequality and equality constraints, respectively. The sets $\Omega \subseteq S$ and $\phi = S - \Omega$ define the feasible and infeasible search spaces, respectively. Usually, the search space S is defined as an n-dimensional rectangle in $\Re^n$ as in (2) (domains of variables defined by their lower and upper bounds):

$$S = \left\{ \mathbf{x} \in \Re^n \ \middle|\ x_j^L \le x_j \le x_j^U, \ \ j = 1,2,...,n \right\} \quad (2)$$

Any point $\mathbf{x} \in \Omega$ is called a feasible solution, otherwise, X is an infeasible solution.

Overview of the FOA and the FA Algorithms

Fruit fly optimization algorithm (FOA)

In this section we describe the basic steps of the original Fruit fly optimization algorithm (FOA) in details as follows [10].

Step 1: Set the main parameters of FOA and give initial location for the fruit fly swarm randomly P

x-axis, y-axis

Step 2: Endow personal fruit fly with random direction for finding food by using:

xi = X- axis + Random value (3)

Yi = y- axis + Random Value

i =1,2,…,m

where m is the swarm size of fruit flies

Step 3: Since the food location cannot be known, calculate the distance ($Dist_i$) of the fruit fly from the origin and find out the smell concentration judgment value (S_i). Suppose that S_i is the reciprocal ($Dist_i$) of as follows:

$$Dist_i = \sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2} \quad (4)$$
$$S_i = 1/Dist_i \quad (5)$$

Step 4: Calculate the smell concentration ($Smell_i$) of the individual fruit fly location by substituting the smell concentration judgment value (S_i) into the smell concentration judgment function (also called Fitness function).

$$Smell_i = \text{Function } (S_i) \quad (6)$$

Step 5: Find out the individual fruit fly with the maximal smell concentration (the maximal value of Smell) among the fruit fly swarm:

$$[best\ smell\ best\ index] = max\ (Smell) \quad (7)$$

Step 6: Keep the maximal smell concentration value and location (x, y) of the best fruit fly. Then, the swarm flies towards that location.

Smell best = Best smell

$$x\text{-}axis = x(best\ index) \quad (8)$$

$$y\text{-}axis = y(best\ index)$$

Enter iterative optimization to repeat the implementation of step 2–6. When the smell concentration is not superior to the previous iterative smell concentration any more, or the iterative number reaches the maximal iterative number, the circulation stops.

Firefly algorithm (FA)

The Firefly algorithm was developed by Xin-She [6] and it is based on idealized behavior of the flashing characteristics of fireflies. For simplicity, we can summarize these flashing characteristics as the following three rules:

a) All fireflies are unisex, so that one firefly is attracted to other fireflies regardless of their sex.

b) Attractiveness is proportional to their brightness, thus for any two flashing fireflies, the less bright one will move towards the brighter one. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If no one is brighter than a particular firefly, it will move randomly.

c) The brightness of a firefly is affected or determined by the landscape of the objective function to be optimized [6].

For simplicity we can assume that the attractiveness of a firefly is determined by its brightness or light intensity which in turn is associated with the encoded objective function. In the simplest case for an optimization problem, the brightness I of a firefly at a particular position X can be chosen as $I(\mathbf{x}) \propto 1/f(\mathbf{x})$. However the attractiveness is relative, it should vary with the distance r_{ij} between firefly i and firefly j. As light intensity decreases with the distance from its source and light is also absorbed in the media, so we should allow the attractiveness to vary with degree of absorption.

Step 1: Attractiveness and light intensity

In the firefly algorithm, there are two important issues: the variation of the light intensity and the formulation of the attractiveness. We know, the light intensity $I(r)$ varies with distance r monotonically and exponentially, that is:

$$I(r) = I_0 e^{-\gamma r^2} \quad (9)$$

Where I_0 the original light intensity and γ is the light absorption coefficient. As firefly attractiveness is proportional to the light intensity seen by adjacent fireflies, we can now define the attractiveness β of a firefly by Equation (10)

$$\beta(r) = \beta_0 e^{-\gamma r^2} \quad (10)$$

Where r is the distance between each two fireflies and β_0 is their attractiveness at r = 0 i.e., when two fireflies are found at the same point of search space. The value of γ determines the variation of attractiveness with increasing distance from communicated firefly.

Step 2: Distance

The distance between any two fireflies *i* and *j* at x_i and x_j respectively, the Cartesian distance is determined by equation (11) where $x_{i,k}$ is the *kth* component of the spatial coordinate of x_i the *ith* firefly and n is the number of dimensions.

$$r_{ij} = \sqrt{\sum_{k=1}^{n} (x_{i,k} - x_{j,k})^2} \quad (11)$$

Step 3: Movement

The firefly i movement is attracted to another more attractive (brighter) firefly *j* is determined by:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i + \beta_0 e^{-\gamma r^2} (\mathbf{x}_j - \mathbf{x}_i) + \alpha(rand - 0.5) \quad (12)$$

Where the second term is due to the attraction while the third term is randomization with α being the randomization parameter and *r and* is a random number generator uniformly distributed in [0, 1].

The Hybrid FOA-FA Algorithm

In this section, the proposed algorithm will be presented in detail. The main reason for developing the FOA-FA is to overcome the drawbacks of the original FOA which cannot handle the negative domain that is the candidate solution cannot be uniformly generated in the problem domain. Also original FOA is easy to fall into premature convergence because the random term of the Equation (3) picks small values within a radius equal to one around the best location.

The methodology of the proposed FOA-FA algorithm consists of two phases. The first one employs the FOA, where a swarm of fruit flies that moves in various directions using the ARM. Therefore, these moves will follow a uniform distribution. The other phase incorporates the FA to update the previous best locations of fruit flies to force FOA jump out of premature convergence, because of its exploitation and exploration ability. Consequently, the hybrid algorithm speeds up the convergence and improves the algorithm's performance. The main steps of the proposed algorithm as follows:

Step 1: Initialize the parameters

The main parameters of the proposed algorithm are the maximum iterationnumber (T),thewarmsize(m) and distance zone of fruit fly *(R)*.

The main parameters of the proposed algorithm are the maximum iteration number (T),thewarmsize(m) and distance zone of fruit fly *(R)*.

Step 2: Initialize the swarm location

Initial fruit fly swarm location is accompanied within the search space, that is, the algorithm assigns a random vector $\mathbf{x}_i - axis = (x_1, x_2, ..., x_n)$ for *i-th* fruit fly as in Equation (13):

$$\mathbf{x}_i - axis = \mathbf{x}^L + rand.(\mathbf{x}^U - \mathbf{x}^L) \quad i = 1,2,...,m. \quad (13)$$

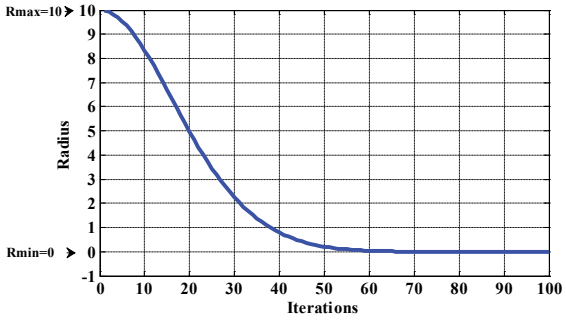


Figure 1: The variation of $\Delta_i(t,R)$ versus the number of iterations.

Where rand is a random value from the uniform distribution on the interval [0, 1].

Step 3: Solution construction using osphresis

The osphresis that introduced in the original FOA (see Equation 3) was picked small values within a radius that equals one. So, we introduce the ARM for enhancing the diversity of solutions and achieving an effective exploration to the whole scope around the fruit flies locations, where the search radius decreases dynamically with iteration number as shown in Figure 1 (i.e., R=100, T=100). Therefore, the process of the solution construction using osphresis is implemented as follows:

$$\mathbf{x}_i = \mathbf{x}_i - axis \pm \varepsilon.\Delta(t,R) \quad (14)$$

$$\Delta_i(t,R) = R\cos\{\frac{\pi}{2}\sin\frac{\pi}{2}[\sin(\frac{\pi}{2}\sin(\frac{\pi t}{T}))]\}, t = 1,2,...,T.$$

Where R is the search radius that determines the extent of the search subspace which we want to cover during the run and ε is a random number from [0, 1]. The random step $\Delta(t,R)$ returns a value in the range *[0, R]* such that the probability of $\Delta(t,R)$ being close to 0 increases as *t* increases. The solution construction using osphresis is demonstrated in Algorithm.

Algorithm 1. The solution construction using osphresis

1. Input : $R = \mathbf{x}^U - \mathbf{x}^L, T, \mathbf{x}_i - axis$

2. set $t = 1, i = 1, \varepsilon = rand(0,1)$

3. $\mathbf{x}_i = \mathbf{x}_i\_axis \pm \varepsilon.\Delta_i(t,R)$

4. if $\mathbf{x}_i\_axis > \mathbf{x}^U$, **set** $\mathbf{x}_i\_axis = \mathbf{x}^U$, **then** $\mathbf{x}_i = \mathbf{x}_i\_axis - \varepsilon.\Delta_i(t,R)$

5. elseif $\mathbf{x}_i\_axis < \mathbf{x}^L$, **set** $\mathbf{x}_i\_axis = \mathbf{x}^L$ **then** $\mathbf{x}_i = \mathbf{x}_i\_axis + \varepsilon.\Delta_i(t,R)$

6. end if

7. $t = t + 1$

8. $i = i + 1$

9. Output: $\mathbf{x}_i \in \Omega$

Step 4: Evaluation

The smell concentration ($Smell_i$) of the individual fruit fly is related to objective function (fitness function). In this step, the fitness

function is evaluated based on the decision variable X as in Equation (15) instead of using the smell concentration judgment value (S_i). Therefore, the best food source with the lowest fitness x_{best} is also found as in Equation (16), i.e., $\mathbf{x}_{best} = \mathbf{arg}(\mathbf{min}_{i=1}^{m} f(\mathbf{x}_i))$.

$$Smell_i = f(x_i),\ i = 1,2,..m \tag{15}$$

$$\mathbf{x}_{best} = \mathbf{arg}(\mathbf{min}_{i=1,2,...,m}(Smell_i)) \tag{16}$$

Step 5: Update the swarm location

In this step, the swarm flies toward the best obtained location using Equation (16). If $\mathbf{X}_{best}$ is better than the current fruit fly swarm location $\mathbf{X}_i$- axis, it will replace the swarm location and become a new one in the next iteration, i.e., $\mathbf{x}_i - axis = \mathbf{x}_{best}$, if $f(\mathbf{x}_{best}) < f(\mathbf{x}_i - axis)$.

Step 6: Crossover operator for updating the osphresis

In this step, the crossover operator is adopted for enhancing the osphresis between fruit flies as follows:

$$\mathbf{x}_i = \beta.\mathbf{x}_i(t) + (1-\beta)\ \mathbf{x}_{best}(t) \tag{18}$$

where $\beta = (1-2\mu)\gamma + \mu$ is the coefficient of modulation, it decides the scope of modulation, $\mu \in [0,1]$ is random numberand isthe extending-modulation operator is specified by the user. The idea of this step is to make the new generation of fruit flies around the best fruit fly, where the new solution not within $x_i(t)$ and $x^{best}(t)$ but the segment may be extended equally on both sides (i.e., 3γ-3 as in Figure 2.

Therefore, the best food source with the lowest fitness $\mathbf{x}_{best}$ is also found as in Equation (16), i.e., $\mathbf{x}_{best} = \mathbf{arg}(\mathbf{min}_{i=1}^{m} f(\mathbf{x}_i))$. If $\mathbf{x}_{best}$ is better than the current fruit fly swarm location X_i-*axis* , it will replace the swarm location and become a new one in the next iteration, i.e., $\mathbf{x}_i - \text{axis} = \mathbf{x}_{best}$, if $f(\mathbf{x}_{best}) < f(\mathbf{x}_i - \text{axis})$.

Step 7: FA phase for updating the diversity

The random term of the Equation (3) makes the fruit flies swarm fluctuates around the so far best solution; consequently, the diversity loss occurs. Under this condition, the best solution might be easily trapped in a local optimum or premature convergence. FA is incorporated to update the previous best locations of fruit flies to force FOA jump out of premature convergence, because of its exploitation and exploration ability. Consequently, the hybrid algorithm speeds up the convergence and improves the algorithm's performance. In the rest of this paper, we will discuss the implementation FA in three steps as follows (Figure 3).

Step 7.1: Initialization: Initialize a swarm of fireflies with the obtained best ant position, where each firefly contains *n* variables (i.e., the position of the i^{th} firefly in the n dimensional search space can be represented as $\mathbf{X}_i = (x_{i1}, x_{i2}, ..., x_{in})$). Furthermore every member of the swarm is characterized by its light intensity (i.e., initialize each firefly with distinctive light intensity. $I_i(x_i)$, $i = 1, 2, \ldots m$

Step 7.2: Light intensity

Calculate the light intensity $I_i(x_i)$, $i = 1, 2, \ldots m$ for each firefly which in turn is associated with the encoded objective function, where for

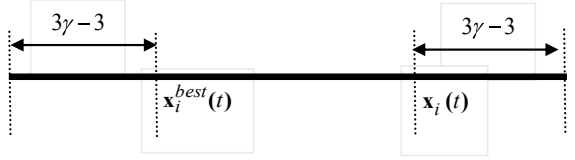


Figure 2: The demonstration of updating the osphresis.

FOA phase

Input: Parameters: T, m, $\beta_0, I_0, \theta, \gamma, \alpha$ and $R = \mathbf{x}^U - \mathbf{x}^L$. Set $t = 0$

Initialization:

for $i = 1,2,...,m$ // generate m food sources $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$

$\mathbf{x}_i - axis = \mathbf{x}^L + rand.(\mathbf{x}^U - \mathbf{x}^L)$

end

$\mathbf{x}_{best}^{(0)} = \mathbf{arg}(\mathbf{min}_{i=1,2,...,m}(Smell_i))$

$\mathbf{x}_i - axis = \mathbf{x}_{best}^{(0)}$

Looping

while $t <= T$ **do**

for $i = 1$ *to* m

$\Delta(t,R) = R\cos\{\frac{\pi}{2}\sin\frac{\pi}{2}[\sin(\frac{\pi}{2}\sin(\frac{\pi t}{T}))]\}$

$\mathbf{x}_i = \mathbf{x}_i - axis \pm \varepsilon.\Delta(t,R)$ // generate new m food sources $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$

if $\mathbf{x}_i\_axis > \mathbf{x}^U$, **set** $\mathbf{x}_i\_axis = \mathbf{x}^U$, **then** $\mathbf{x}_i = \mathbf{x}_i\_axis - \varepsilon.\Delta_i(t,R)$

else if $\mathbf{x}_i\_axis < \mathbf{x}^L$, **set** $\mathbf{x}_i\_axis = \mathbf{x}^L$ **then** $\mathbf{x}_i = \mathbf{x}_i\_axis + \varepsilon.\Delta_i(t,R)$

if $\mathbf{x}_i \notin \Omega$, **then** $t = t+1$

$i = i+1$

end for

$Smell_i = f(\mathbf{x}_i), i = 1,2,...,m$ // evaluation.

$\mathbf{x}_{best} = \mathbf{arg}(\mathbf{min}_{i=1,2,...,m}(Smell_i))$

if $f(\mathbf{x}_{best}) < f(\mathbf{x}_i - axis)$ **then,** $\mathbf{x}_i - axis = \mathbf{x}_{best}$

if $f(\mathbf{x}_i - axis) < f(\mathbf{x}_{best})$ **then ,** $\mathbf{x}_i - axis = \mathbf{x}_i - axis$

for $i = 1,2,...,m$ // updating the osphresis

$\mathbf{x}_i' = \beta.\mathbf{x}_i(t) + (1-\beta).\mathbf{x}_{best}(t)$

end

$Smell_i = f(\mathbf{x}_i'), i = 1,2,...,m$ // evaluation.

$\mathbf{x}_{best}' = \mathbf{arg}(\mathbf{min}_{i=1,2,...,m}(Smell_i))$

if $f(\mathbf{x}_{best}') < f(\mathbf{x}_i - axis)$ **then,** $\mathbf{x}_i - axis = \mathbf{x}_{best}'$

if $f(\mathbf{x}_i - axis) < f(\mathbf{x}_{best}')$ **then ,** $\mathbf{x}_i - axis = \mathbf{x}_i - axis$

FA phase

Initialization:

for $i = 1,2,...,m$ // generate m locations for fireflies $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$

$\mathbf{x}_i = \mathbf{x}^L + rand.(\mathbf{x}^U - \mathbf{x}^L)$

end

$I(\mathbf{x}_i) = f(\mathbf{x}_i), i = 1,2,...,m.$ // evaluation.

while $t <= T$ **do**

$\alpha_{t+1} = \alpha_t.\theta^{(1-t/T)}$

for $i = 1$ *to* m **do**; **for** $j = 1$ *to* m **do**.

if $I_i > I_j$ **then** $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \beta_0 e^{-\gamma r^2}(\mathbf{x}_j(t) - \mathbf{x}_i(t)) + \alpha(rand - 0.5)$

end for j ; **end for** i

if $\mathbf{x}_i > \mathbf{x}^U$, **then** $\mathbf{x}_i = \mathbf{x}^U$

else if $\mathbf{x}_i < \mathbf{x}^L$, **then** $\mathbf{x}_i = \mathbf{x}^L$

$I(\mathbf{x}_i) = f(\mathbf{x}_i), i = 1,2,...,m.$ // evaluation.

$\mathbf{x}_{best}'' = \mathbf{arg}(\mathbf{min}_{i=1,2,...,m} I(\mathbf{x}_i))$

if $f(\mathbf{x}_{best}'') < f(\mathbf{x}_i - axis)$ **then,** $\mathbf{x}_i - axis = \mathbf{x}_{best}''$

if $f(\mathbf{x}_i - axis) < f(\mathbf{x}_{best}'')$ **then ,** $\mathbf{x}_i - axis = \mathbf{x}_i - axis$

end Looping

Output: $\mathbf{x}_i - axis$

Figure 3: The pseudo code of the proposed FOA-FA algorithm.

minimization problem the brighter firefly represents the minimum value for $I(x)$.

Step 7.3: Attractiveness and movement

In the firefly algorithm, the attractiveness of a firefly is determined by its light intensity which in turn is associated with the encoded objective function and the distance r_{ij} between firefly i and firefly j. In addition, light intensity decreases with the distance from its source, and light is also absorbed in the media, so we should allow the attractiveness to vary with the degree of absorption. As a firefly's attractiveness is proportional to the light intensity seen by adjacent fireflies, we can now define the attractiveness β of a firefly by Equation (10). On the other hand, the movement of a firefly i is attracted to another more attractive (brighter) firefly J is determined by Equation (13).

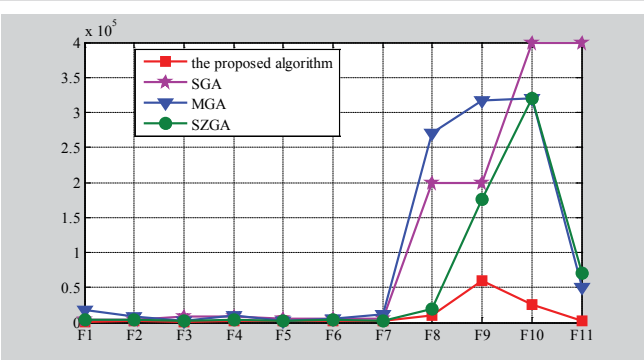


Figure 4: The graphical representation for the evaluation of metric.

$$\mathbf{x}_i = \mathbf{x}_i + \beta_0 e^{-\gamma r^2}(\mathbf{x}_j - \mathbf{x}_i) + \alpha_1(rand - 0.5)$$

Since each member of the swarm explores the problem space taking into account results obtained by others, therefore the randomization term may moves the fire fly to lose its best location, so we introduce a modification on the randomization term that makes the fireflies approached from the optimum. A further improvement on the convergence of the algorithm is to vary the randomization parameter α_1 so that it decreases gradually as the optima are approaching. For example, we can use

$$\alpha_{t+1} = \alpha_t \theta^{(1-\frac{t}{Tc})}, t = 1,2,...,Tc. \quad (14)$$

Where T_c is the maximum number of generations for FA, and θ ϵ [0, 1] is the randomness reduction constant.

The basic steps of the proposed FOA-FA algorithm can be summarized as the pseudo code shown in Figure 4.

Results and Discussions

In this section, performance of the proposed FOA-FA algorithm is tested on eleven benchmark problems, which are known from the literature and reported in Young [16] and two engineering applications [17,18]. The solutions of the proposed FOA-FA are compared with the prominent algorithms that reported in Young [16-18]. The test problems are listed in Table 1. The algorithm is coded in MATLAB 7, which has an Intel Core I 5 (1.8 GHz) processor and 4 GB RAM memory.

Test problems	Dimension	Domain	Optimum value
$F_1 = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	2	$x_1, x_2 \in [-1.28, 1.28]$	0
$F_2 = [\cos(2\pi x_1) + \cos(2.5\pi x_1) - 2.1] * [2.1 - \cos(3\pi x_2) + \cos(3.5\pi x_2)]$	2	$x_1, x_2 \in [-1,1]$	-16.09172
$F_3 = \left[0.002 + \sum_{j=1}^{25}(j + \sum_{i=1}^{2}(x_i - a_{ij})^6)^{-1}\right]^{-1}$	2	$x_1, x_2 \in [-65.536, 65.536]$	0.9980
$a = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & -16 & -16 & -16 & 0 & 0 & 0 & 0 & 0 & 16 & 16 & 16 & 16 & 16 & 32 & 32 & 32 & 32 & 32 \end{bmatrix}$			
$F_4 = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	2	$x_1 \in [-5,10], x_2 \in [0,15]$	0.39789
$F_5 = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(4x_2^2 - 4\right)x_2^2$	2	$x_1 \in [-3,3], x_2 \in [-2,2]$	-1.03163
$F_6 = \left(1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right) * \left(30 + (2x_1 - 3x_2)^2\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right)$	2	$x_1, x_2 \in [-2,2]$	3
$F_7 = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right) * \left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$	2	$x_1, x_2 \in [-10,10]$	-186.73091
$F_8 = \sum(a x_1^{c_1} x_2^{c_2} - f_{simulated})^2$	3	$a, c_1, c_2 \in (-\infty, \infty)$	8.01276
$f_{simulated}$ / x_1 / x_2: 2.122 / 5 / 10; 9.429 / 3 / 1; 23.57 / 0.6 / 0.6; 74.25 / 0.1 / 2.0; 6.286 / 3.0 / 1.8			
$F_9 = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2)$	4	$x_i \in [-10,10], i = 1,2,...,4$	0
$F_{10} = \sum_{i=1}^{19}\left[(x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)}\right]$	20	$x_i \in [-1,4], i = 1,2,...,20$	0
$F_{11} = (\frac{\pi}{20})\left[10\sin^2(\pi x_1) + \sum_{i=1}^{19}\left((x_i - 1)^2(1 + 10\sin^2(\pi x_{i+1}))\right) + (x_{20} - 1)^2\right]$	20	$x_i \in [-10,10], i = 1,2,...,20$	0

Table 1: Test problems.

On the other hand, extensive experimental tests were conducted to see the effect of different values on the performance of the proposed algorithm. Based upon these observations, the following parameters have been set as in Table 2.

The performance evaluation metrics

Our hybrid algorithm was tested on set of the test problems in engineering domain. Each of the test problems was run thirty times independently, with different seeds. In order to evaluate the closeness between the obtained optimal solution and the theoretical optimal solution, as well as measuring the number of evaluations for the proposed algorithm and the existing algorithms, we used two different metrics.

The speed: The speed with which the optimum is obtained is a criterion for examining the performance. For the proposed FOA-FA to be competitive, it is important to be able to compare its speed with the other existing algorithms. As time is a relative notion depending on the computer used, we preferred to measure the speed by counting the number of evaluations of the tested problem. On the one hand, this criterion is independent from the type of computer used, and on the other hand the evaluation of the function is the operation taking the longest time in the program.

Number of iteration (*T*)	200
The swarm size (m)	100
Initial attractiveness (β0)	1
The light absorption coefficient (γ)	1
The randomness reduction constant (Θ)	0.96
Randomization parameter (α)	0.95

Table 2: Parameters settings.

The error: So the test for closeness, we used the evaluation of the error between the result given by the algorithm and the exact value of the optimum, for each test problem. The relative error was used each time it was possible:

$$E_{relative} = \frac{\left|F_{obtained} - F_{optimum}\right|}{F_{optimum}} \quad (20)$$

When the optimum to reach was 0, it was no longer possible to use this expression so we calculated the absolute error:

$$E_{relative} = \left|F_{obtained} - F_{optimum}\right| \quad (21)$$

Table 3 compares the performance of the proposed FOA-FA, where the better values in the comparison are bold in font type. On the other hand, a comparison is made with the prominent algorithms

	FOA-FA						Compared algorithms		
	Function value		Number of function evaluation(NFE)		Time(s)		Name	Function value	NFE
F (**x**)	Best	Average	Best	Average	Best	Average			
F_1	**0**	1.792475E-20	1500	1500	0.611614	0.629180	SGA MGA SZGA	0.44108E-5 0.40842E-5 0.29802E-7	4000 18182 4000
F_2	**-16.091720006**	-16.09172	2000	2000	0.307335	0.323088	SGA MGA SZGA	-16.09171 -16.09171 -16.09172	4000 8000 4000
F_3	0.9980	0.9980	1500	1700	0.756351	0.766544	SGA MGA SZGA	0.9980 0.9980 0.9980	8000 2000 2000
F_4	**0.39785**	0.3978873	2500	2500	0.397967	0.401413	SGA MGA SZGA	0.39789 0.39789 0.3979	8000 10000 4000
F_5	-1.03163	-1.0316284	1600	2000	0.129138	0.134047	SGA MGA SZGA	-1.03163 -1.03163 -1.03163	6000 3000 3000
F_6	3	3	2500	2500	0.701674	0.716377	SGA MGA SZGA	3.0001 3.0001 3	6000 5000 4000
F_7	-186.73091	-186.730908	2300	2500	4.8226	4.8243	SGA MGA SZGA	-186.72744 -186.72825 -186.73091	6000 12000 3000
F_8	8.01276	8.0137	10000	10000	19.2651	19.6132	SGA MGA SZGA	8.01359 8.01293 8.01276	200000 270270 20000
F_9	**3.6284E-7**	3.6284E-7	60000	60000	18.8068	18.6979	SGA MGA SZGA	1.63594 0.44407E-2 0.13074E-5	200000 317460 175440
F_{10}	**0**	7.5684E-17	25000	24500	4.786236	4.799961	SGA MGA SZGA	0.15029E-1 0.13565E-1 0.25422E-7	400000 320000 320000
F_{11}	**1.3022E-15**	9.4984E-7	2500	2500	0.957504	15.0280	SGA MGA SZGA	0.89927 0.16851 0.23033E-3	400000 49855 70000

Table 3: The comparison of solution quality.

from the literature. The test problems have been solved by FOA-FA for 30 times. The starting values of the variables for each problem were selected randomly for all runs from the solution space. The results found by FOA-FA such as the best and average function value, numbers offunction evaluation (NFE) and solution time in seconds have been recorded in Table 3, whereas for the other algorithms only the function value and numbers of function evaluation are given because the solution times, the best and average function value for some algorithms not given. First, we start by comparing the optimum solution of the given the test problems for the proposed FOA-FA algorithm and the existing algorithms. Based on the obtained results that demonstrated in Table 3 we notes that, the proposed FOA-FA is successful finding the optimum solution for the test problems F2 and F4 better than both the theoretical optimum solution and the obtained by other existing algorithms. Also the proposed FOA-FA algorithm outperforms the existing algorithms for functions F_1, F_6, F_7, F_8, F_9, F_{10} and F_{11} but equals the SZGA for the test problems F_6, F_7 and F_8. Second, as shown from Table 4, the proposed algorithm speeds up the convergence through inclusion of the lowest evaluations, where the proposed algorithm elapsed number of evaluations less than the other existing algorithms for finding the global optimum as in Figure 4, that is the proposed algorithm is the faster than the existing algorithms.

Finally, the last performance metric is the relative error or the absolute error, where as depicted from Table 4, the absolute error by using the proposed algorithm is better than the other exists algorithms for all test problems except SZGA for the test problem F_4 and also the relative error is superior to the other exists algorithms for all test problems except SZGA for the test problem F_4 The SZGA seems to better than the proposed algorithm for the test problem F_4 this because the error is calculated related to the theoretical optimum. In fact, the proposed algorithm outperforms the theoretical optimum for the test problem F_4, where if the relative error and the absolute error are calculated related to the obtained optimum by our algorithm we find it 0.12567 E-5 and 0.5 E-6 respectively. Therefore, the simulation results prove superiority of the proposed algorithm to those reported in the literature, where it is completely better than the other algorithm

Design optimization problems

The design optimization problems are a very important research area in engineering studies because real world design problems require the optimization of relevant engineering applications. In terms of robustness and efficiency of the available methods, these methods are still in need of improvements. Therefore, to meet the ever increasing demands in the design of the electrical devices, the proposed FOA-FA has been substantiated its capability for solving these applications. Thus, the validity of the proposed algorithm has been proved superiority by using two engineering applications, the linear synchronous motor (LSM) and air-cored solenoid [17,18].

Problems	Absolute error			Relative error		
	Proposed algorithm	Compared algorithms		Proposed algorithm	Compared algorithms	
F_1	0	SGA MGA	0.44108E-5 0.40842E-5	-	SGA MGA	- -
		SZGA	0.00298E-5		SZGA	-
F_2	0.6E-8	SGA MGA	0.99999E-5 0.99999E-5	-0.37286E-9	SGA MGA	-0.62143E-6 -0.62143E-6
		SZGA	0		SZGA	0
F_3	0	SGA MGA	0 0	0	SGA MGA	0 0
		SZGA	0		SZGA	0
F_4	0.4E-4	SGA MGA	0 0	0.10053E-3	SGA MGA	0 0
		SZGA	0.99999E-5		SZGA	0.251325E-4
F_5	0	SGA MGA	0 0	0	SGA MGA	0 0
		SZGA	0		SZGA	0
F_6	0	SGA MGA	0.1E-3 0.1E-3	0	SGA MGA	0.3333E-4 0.3333E-4
		SZGA	0		SZGA	0
F_7	0	SGA MGA	0.003469 0.002659	0	SGA MGA	-0.18582E-4 -0.14245E-4
		SZGA	0		SZGA	0
F_8	0	SGA MGA	0.83E-3 0.17E-3	0	SGA MGA	0.10358E-3 0.021216E-3
		SZGA	0		SZGA	0
F_9	0.36284 E-6	SGA MGA	1.63594 0.44407E-2	-	SGA MGA	- -
		SZGA	0.13074E-5		SZGA	-
F_{10}	0	SGA MGA	0.15029E-1 0.13565E-1	-	SGA MGA	- -
		SZGA	0.25422E-7		SZGA	-
F_{11}	0.13022E-14	SGA MGA	0.89927 0.16851	-	SGA MGA	- -
		SZGA	0.23033E-3			-

Table 4: The performance assessment.

Design of LSM: The linear synchronous motor (LSM) operates on the same working principle as that of a permanent magnet rotary D.C. motor [17]. As in a rotary motor there are two parts in a LSM, one is the set of permanent magnets and the other is the armature that has conductors carrying current. The permanent magnets produce a magnetic flux perpendicular to the direction of motion. The flow of current is in the direction perpendicular to both the direction of the motion and the direction of the magnetic flux.

To validate the proposed algorithm, it is employed to optimize the geometrical design of the linear electric actuator problem as is shown in Figure 5(a) and 5(b). The objective function is to maximum force subject to some of constraints on heat, radius, saturation, demagnetization and maximum force constraint. There are four design variables: the current in each slot, x_1, the dimensions of the slot, and x_2 x_3, and the height of the magnet, x_4. The mathematical formulation of the objective function is described as follows:

$$\max F_{12}(\mathbf{x}) = 653.45x_1x_4(0.02825 + x_4 + x_2/2)/(x_4 + 0.00117)$$

subject to :

$$g_1(\mathbf{x}) = 1.92 * 10^{-5} x_1^2(0.02825 + x_4 + x_2/2)/x_2x_3 - -4000 \le 0$$

$$g_2(\mathbf{x}) = x_4 + x_2 - 3x_3 - 0.0266 \le 0$$

$$g_3(\mathbf{x}) = 1.3x_4/(0.00117 + x_4) + 165.13x_3 - 1.5 \le 0$$

$$g_4(\mathbf{x}) = 6.28 * 10^{-7} x_1(0.00105 + x_4)/x_4(x_4 + 0.001) - 1.17 \le 0$$

$$g_5(\mathbf{x}) = 15000 - 653.45x_1x_4(0.02825 + x_4 + x_2/2)/(x_4 + 0.00117) \le 0$$

$$0 \le x_1 \le 1028A,\ 0 \le x_2, x_4 \le 0.08m, 0 \le x_3 \le 0.009m.$$

The obtained result for design of LSM is demonstrated in Table 5, by giving the best, worst and mean values for the objective function. Also Figure 6 depicts the relation between the maximum force and number of iterations for FOA before applying FA and after Applying FA (i.e., the hybrid algorithm FOA-FA). The results of the proposed FOA-FA is superior to those obtained using the Deshpande [17].

Design of an air-cored solenoid: The optimization problem of a coreless solenoid with rectangular cross section $x_1 \times x_i$ and a mean radius x_3Figure 5(b) is tackled from Barba [18]. This problem can be formulated as: maximize the inductance L(x_1, x_2, x_3) and satisfy some of constraints for the given length $k_1 = 10\ m$ and $k_2 = 10^{-6}$ of the current carrying wire. In order to simplify the analysis, two variables, x_1 and x_2, are considered. Correspondingly, the optimization problem is simplified as:

Methods	FOA-FA	FOA	Deshpande [13]
Best objective function value	17705	1.6239 E4	16192
Worst objective function value	12139	15440	-
Mean objective function value	1.7043 E4	1.6198 E4	-
Median objective function value	17703	1.6239 E4	-
NFE	4000	4500	-
Time (second)	3.449	0.3812	-

Table 5: Optimal results of the LSM problem.

$$\max F_{13}(\mathbf{x}) = 31.49\frac{k_1^2}{4\pi^2 x_2}\Big/(9 + 6\frac{x_1}{x_2} + 5\frac{k_1k_2}{\pi x_1x_2^2}),$$

subject to :

$$g_1(\mathbf{x}) = \frac{\pi x_1^2 x_2}{4} + \frac{k_1^2k_2^2}{4\pi x_1^2 x_2} + \frac{k_1k_2}{2} \le 0.0094$$

$$g_2(\mathbf{x}) = x_1 - \sqrt{\frac{k_1k_2}{4\pi x_2}} > 0,\ \ , x_1 \in [0, 0.1], x_2 \in [0, 0.3]$$

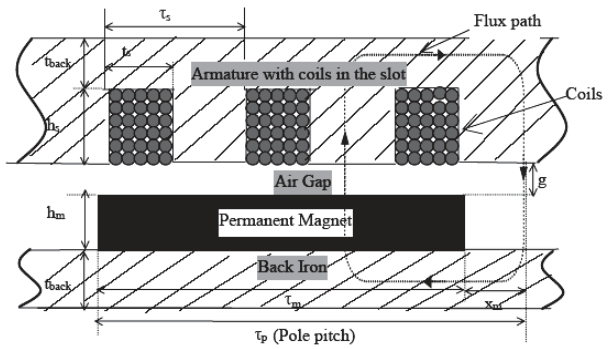
Table 6 demonstrates the result for design of air-cored solenoid, by giving the best, worst and mean values for the objective function. On the other hand, Figure 7 depicts the relation between the maximum inductance against the number of iterations for FOA before applying FA and after Applying FA (i.e., the hybrid FOA-FA). The result of the proposed FOA-FA is superior to those obtained by [18].

Figure 8 shows the graphical representation of the optimum solution for the LSM and air cored solenoid. It is obvious that the algorithm outperforms the other algorithms for the both problems.

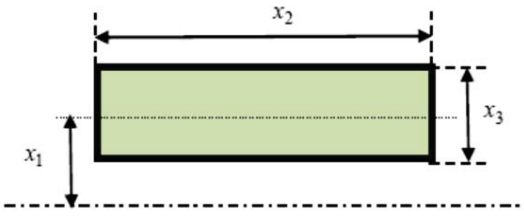
We concluded that the integrated algorithm of FOA with FA has improved the quality of the founded solutions and also it guarantees the faster converge to the optimal solution. The FOA is presented in the

Methods	FOA-FA	FOA	Barba [14]
Best objective function value	258.0607	258.0291	257.197
Worst objective function value	250.0581	244.3131	-
Mean objective function value	257.7696	257.5935	-
Median objective function value	257.9143	258.0084	-
NFE	3300	4477	-
Time (second)	1.2345	0.4378	-

Table 6: Optimal results of the air-cored solenoid.



(a). Cross-section of approximately a one pole pitch long section of LSM



(b). Cross section of the solenoid and design variables

Figure 5: The configurations of two designs.

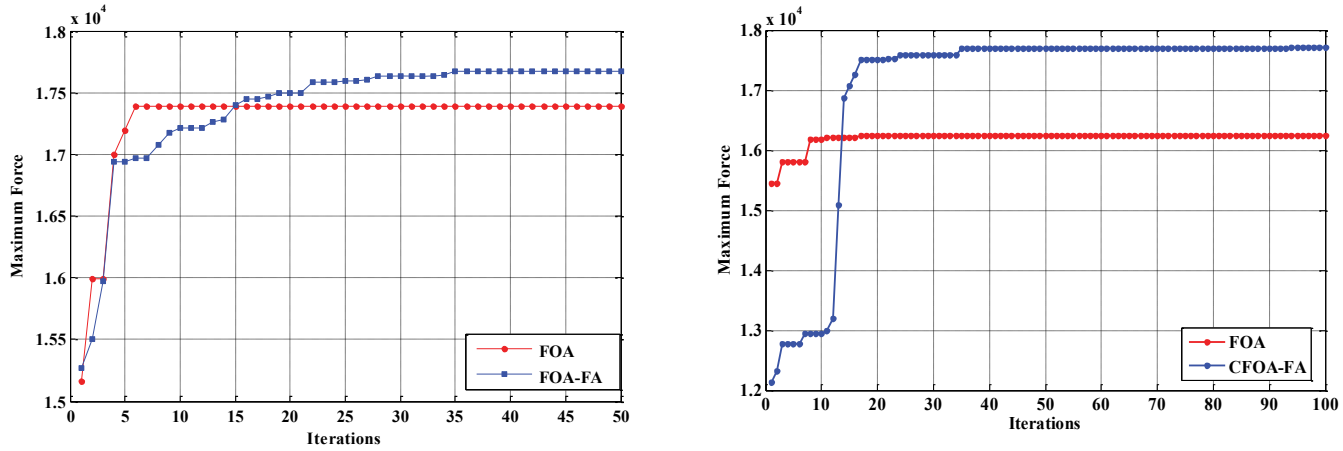


Figure 6: Maximum force by using FOA and FOA-FA.

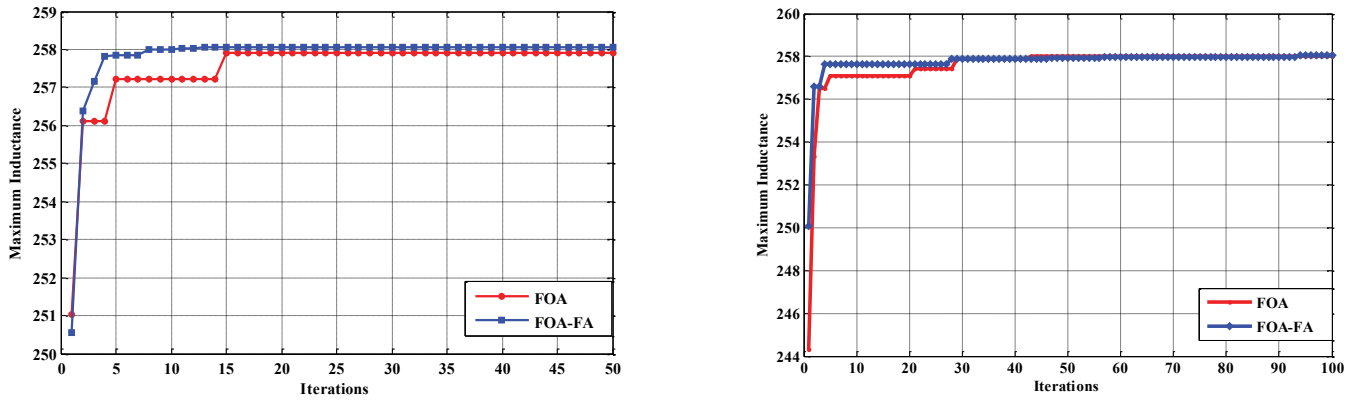
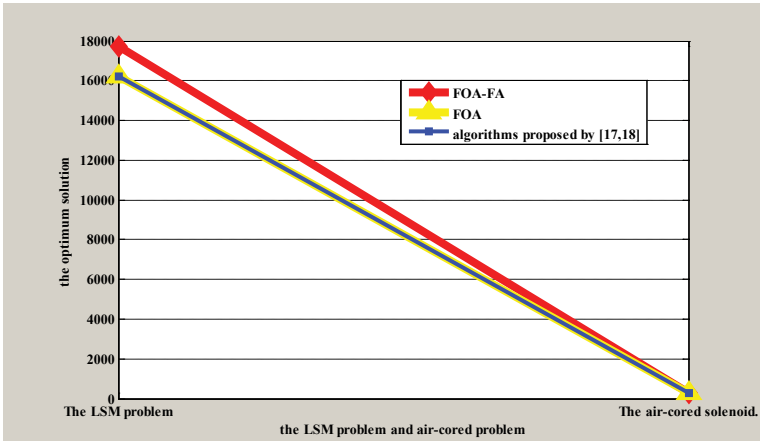


Figure 7: Maximum inductance by using FOA and FOA-FA.



Figuge 8: The graphical representation of the optimum solution.

first stage to provide the initial solution to optimization problem (close to the optimal solution as possible) followed by FA to enhance the quality of the solution. However, because of its random behavior, FOA may suffer from slow convergence. So we introduced the neighborhood search via adaptive radius mechanism $\Delta(t,R)$ that returns a value in the range $[0, R]$. On the other hand, to reach a quick and closer result to optimal solution, and to improve the efficiency of the FOA, hybridization between FOA and FA was implemented.

In this subsection, a comparative study has been carried out to assess the performance of the proposed FOA-FA algorithm concerning the hybridization, closeness to optimal solution and computational time. On

the first hand, pure algorithms suffer from reach to an optimal solution in a reasonable time. Also, the sinking into premature convergence is inevitable in pure algorithms. Consequently, our hybridization algorithm has twofold feature, solving NLPPs in a reasonable time and avoiding the sinking into the premature convergence. On the other hand, the proposed FOA-FA algorithm is highly competitive when compared with the other existing methods in term of calculating the relative, absolute error and number of evolutions. Accordingly, it provides the facility to search in the negative and the nonnegative orthant through adaptive radius mechanism. Finally, the feasibility of using the proposed algorithm to handle NLPPs has been empirically approved.

Conclusions

A novel hybrid algorithm named, FOA-FA for solving the nonlinear programming problems and engineering applications is proposed. The proposed FOA-FA algorithm employed the merits of both fruit fly optimization algorithm (FOA) and firefly algorithm (FA) and consists of two phases. The first one employs a variation on original FOA employing a new adaptive radius mechanism for exploring the whole scope around the fruit flies locations to overcome the drawbacks of original FOA, which has been an continues to be a fruitful paradigm for designing effective the nonnegative orthant algorithms. Moreover, the premature convergence of original FOA degrades its performance by reducing its search capability, leading to a higher probability of being trapped to a local optimum. Therefore, the second phase incorporates the FA to update the previous best locations of fruit flies to force FOA jump out of premature convergence, because of its strong searching ability. The hybrid algorithm speeds up the convergence and improves the algorithm's performance.

A careful observation will reveal the following benefits of the proposed optimization algorithm.

1) It can efficiently safeguard the solution from sinking into the premature convergence through incorporating the FA.

2) It can overcome the lack of the exploration of the FOA which roaming in the nonnegative orthant by introducing the adaptive radius mechanism.

3) It emphasizes the diversity of solutions by incorporating the FA that enriches the exploratory capabilities of the proposed algorithm.

4) It competitive when compared with the other existing algorithm.

5) It can find can find the global minimum for the problems very efficiently.

6) It can accelerate the convergence and improves the algorithm's performance through elapsed low computational time.

The future work will be focused on three directions: (i) the application of FOA- FA to real world problems; (ii) the extension of the method to solve the multi-objective problems; and (iii) another applications in engineering.

References

1. Deep K, Dipti (2008) A self-organizing migrating genetic algorithm for constrained optimization. Applied Mathematics and Computation 198: 237-250.

2. He Q, Wang L (2007) A hybrid particle swarm optimization with a feasibility – based rule for constrained optimization. Applied Mathematics and Computation 186: 1407-1442.

3. Becerra RL, Coello CAC (2006) Cultured differential evolution for constrained optimization. Computer Methods in Applied Mechanics and Engineering 195: 4303-4322.

4. Mousa MM, El-Wahed WFA, Rizk-Allah RM (2011) A Hybrid Ant Colony Optimization Approach Based Local Search Scheme for Multiobjective Design Optimizations. Journal of Electric Power Systems Research 81: 1014-1023.

5. Yang XS (2009) Firefly algorithms for multimodal optimization, In Stochastic Algorithms: Foundation and Applications. SAGA 2009, Lecture Notes in Computer Sciences 5792: 169-178.

6. Yang XS (2008) Nature-Inspired Metaheuristic Algorithms. Luniver Press.

7. Rizk-Allah RM, Elsayed ZM, El-Sawy EA (2013) Hybridizing ant colony optimization with firefly algorithm for unconstrained optimization problems. Applied Mathematics and Computation 224: 473-483.

8. El-Sawy A, Zaki EM, Rizk-Allah RM (2013) A Novel Hybrid Ant Colony Optimization and Firefly Algorithm for Solving Constrained Engineering Design Problems. Journal of Natural Sciences and Mathematics 6: 122.

9. El-Sawy A, Zaki EM, Rizk-Allah RM (2013) Novel hybrid ant colony optimizationand firefly algorithm for multi-objective optimization problems. International Journal of Mathematical 4: 152-161.

10. Pan WT (2012) A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowledge-Based Systems 26: 6-74.

11. Li H, Guo S, Li L, Sun J (2013) A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm. Knowl-Based Syst 37: 378-387.

12. Lin SM (2013) Analysis of service satisfaction in web auction logistics service using a combination of fruit fly optimization algorithm and general regression neural network. Neur Comput Appl 7: 459-465.

13. Han J, Wang P, Yang X (2012) Tuning of PID controller based on fruit fly optimization algorithm, in: International Conference on Mechatronics and Automation (ICMA) pp: 409-413.

14. Wang L, Zheng XL, Wang SL (2013) A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem. Knowl-Based Syst 48: 17-23.

15. Bazaraa MS, Shetly CM (1979) Nonlinear Programming Theory and Algorithms. Wiley, New York.

16. Young-Doo K, Soon-Bum K, Seung-Bo Y, Jae-Yong K (2003) Convergence enhanced genetic algorithm with successive zooming method for solving continuous optimization problems. Computers and Structures 81: 1715-1725.

17. Deshpande AD (2002) California Linear Drives, Inc, A Study Of Methods To Identify Constraint Dominance In Engineering Design Problems. MS Thesis, Mechanical and Industrial Engineering Department, University of Massachusetts, Amherst.

18. Barba PB, Farina M, Savini A (2001) An improved technique for enhancing diversity in Pareto evolutionary optimization of electromagnetic devices. COMPEL 20: 482-496.

Citation: Allah RMR (2016) Hybridization of Fruit Fly Optimization Algorithm and Firefly Algorithm for Solving Nonlinear Programming Problems. Int J Swarm Intel Evol Comput 5: 134. doi: 10.4172/2090-4908.1000134