International Journal of Artificial Intelligence,
ISSN 0974-0635; Int. J. Artif. Intell.
Spring (March): 2015, Volume 13, Issue Number 2
Copyright © 2015 by IJAI (CESER PUBLICATIONS)

Fish Swarm Search Algorithm: A New Algorithm for Global Optimization

Danial Yazdani[1], **Sarvenaz Sadeghi-Ivrigh**[2], **Donya Yazdani**[3], **Alireza Sepas-Moghaddam**[4], and **Mohammad Reza Meybodi**[5]

[1] Young Researchers and Elite Club, Mashhad Branch,
Islamic Azad University, Mashhad, Iran
Email : d_yazdani@mshdiau.ac.ir

[2] Department of Computer, Buinzahra Branch, Islamic Azad University, Buinzahra, Iran
Email: sarvenaz.sadeghi@gmail.com

[3] Science and Reserach Branch, Islamic Azad University, Qazvin, Iran
Email: donya.yazdani@qiau.ac.ir

[4] Department of Pattern Recognition, Pattern Research Center (PRC), Alborz, Iran
Email: sepasmoghaddam@iranprc.org

[5] Soft Computing Laboratory, Computer Engineering and Information Technology Department,
Amirkabir University of Technology, Tehran, Iran
Email: mmeybodi@aut.ac.ir

ABSTRACT

Optimization is amongst the most significant problems in mathematics and sciences and many researchers are investigating different aspects of this problem. In this paper, a novel algorithm has been proposed for optimization in continuous static environments based on the individual and social behaviors of fish in their swarms. The proposed algorithm, so called Fish Swarm Search Algorithm (FSSA), is a population-based algorithm that can considered among swarm intelligence, evolutionary and meta-heuristic approaches. In the proposed algorithm, there is a population of fish in which every individual fish moves toward better positions in the problem space by following better members of the population and performing a random search in the individual space. The proposed algorithm involves several advantages i.e. better intelligibility, simplicity, high convergence rate, high reliability, high balance between exploration and exploitation, and maintaining diversity in the swarm. There is only one parameter, namely population size, which needs to be initialized in the proposed algorithm in order to start the optimization process, which results in a considerable simplicity of the proposed. The proposed algorithm has been compared with nine other well-known algorithms in this domain on thirty benchmark functions with Unimodal, Multimodal, Shifted and Rotated characteristics. The experimental results and analysis reveals the superiority of the proposed method, compared to other comparative studies.

Keywords: Fish swarm Search Algorithm, swarm intelligence, meta-heuristic methods, natural inspired algorithms.

Mathematics Subject Classification: 68T01, 68T20

Computing Classification System: I.2

1. INTRODUCTION

Optimization problem play an important role in mathematics and sciences, where it can be observed in many real-world applications. The goal in most of optimization problems is to discover an optimum for a particular cost functions. Since optimization problems are commonly complicated, low-cost approaches are needed in this domain. Some of these algorithms are based on population. These population-based algorithms present a computing method that utilizes an iterative process for improving the solution until it reaches a termination state. An appropriate population based method is able to establish a trade-off between exploration and exploitation. On the other hand, a weak optimization method suffers from poor balance between exploration and exploitation, which leads to premature convergence, trapping in a local optimum, and stagnation. One of the most well-known population based algorithms are swarm intelligent approaches.

An aggregation of animals e.g. fish swarms and insect colonies is called swarm, where each swarm performs particular social behavior. Each individual in a swarm acts unsupervised based on its knowledge about the environment. The local rules among a swarm define swarm intelligence, where it has no dependency to the global patterns and interactions between self-organized members. This matters leads to effective utilization of the environment and resources. One of a prominent feature of a swarm system is self-organization, where global level response is resulted from low level interactions. Consequently, these models have been utilized to propose computing methods in order to solve different real-world problems.

In this paper, a novel algorithm has been proposed for global optimization in stationary and continuous environments based on individual and social behaviors of fish in their swarms. This algorithm calls Fish Swarm Search Algorithm (FSSA) and is population-based algorithm. FSSA can be also considered among swarm intelligence algorithms, evolutionary algorithms, and meta-heuristic approaches. In the proposed algorithm, the members of the populations, i.e. the fish, search the space in order to find more food. Each fish in a swarm moves toward the better fish and performs a local search in its surrounding environment in order to move toward better positions. In the proposed algorithm, the fish maintain a high diversity in the swarm by repositioning in the swarm according to the central position. Thus, the swarm’s ability to find better peaks and move from local optimums is increased. This algorithm is designed with the fewest parameter settings. The experiments have been done on thirty standard benchmark functions with different characteristics, such as unimodal, multimodal, shifted, and rotated. The performance of the proposed algorithm has been tested and compared on benchmark functions in comparison with nine other well-known algorithms including modified artificial fish swarm algorithm with communication behavior(CM-AFSA) (Tsai and Lin, 2011), global-best particle swarm optimization(GPSO) (Shi and Eberhart, 1998), local-best particle swarm optimization(L-PSO) (Keneddy and Mondes, 2002), adaptive inertia weight particle swarm optimization(AIW-PSO) (Nickabadi et al., 2011), global-best harmony search (Omran and Mahdavi, 2007), imperialist cognitive Improved algorithm(I-ICA) (Shirkouhi et al., 2010), gravitational search algorithm(GSA) (Rashedi et al., 2009), modified artificial bee colony algorithm (M-ABC) (Akay and Karaboga, 2012) and shuffled frog leaping algorithm(SFLA) (Eusuff and Lansey, 2003).

The rest of the paper is organized as follows: in section two a brief review on swarm intelligence algorithms is explained. Section three in dedicated to present the proposed algorithm in details. The results of the experiments are illustrated and analyzed in section four, and the final section concludes the paper.

2. A BRIEF REVIEW ON SWARM INTELLIGENCE ALGORITHMS

Several swarm intelligence algorithms have been proposed by researchers so far, and these algorithms have been applied in many different applications (Yang et al., 2010), (Nolle, et,al., 2005), (Precup, et,al., 2011), (Saha, et,al., 2013), (Ali, et,al., 2013). The proposed algorithm is also one of the meta-heuristic population-based swarm-intelligence optimization algorithms. In these algorithms, there exists a population whose members cooperate in the problem space in order to find better positions. The rest of this section will focus on some of the well-known swarm intelligence algorithms.

Particle Swarm Optimization (PSO) is the most well-known swarm intelligence algorithm proposed by Kennedy and Eberhart (Kennedy and Eberhart, 1995) in 1995. This algorithm simulates the fish and birds behaviors. Many different versions of this algorithm have been proposed so far (Poli et al., 2007), (Li et al., 2011), (Liu et al., 2012), (Zhan et al. 2011). PSO algorithm has been used in many different applications, such as optimization in large-scale environments (Li and Yao, 2012), optimization in discrete environments (Zhang et al., 2010), optimization in dynamic environments (Yang and Li, 2010), (Yazdani et al., 2013), (Yazdani et al., 2013), parameter optimization (Kasabov and Hamed, 2011), job shop scheduling (Jiao and Yan, 2011), damage detection (Kang et al., 2012), function optimization (Li and Liang, 2011), feature selection (Chen et al., 2012), image segmentation (Yazdani et al., 2012).

Artificial bee colony algorithm (ABC) algorithm was proposed by Karaboga in 2005 (Karaboga, 2005). This algorithm uses the bee behavior in searching for nectar (Karaboga and Akay, 2009). It has been used in various applications such as constrained optimization problems (Karaboga and Akay, 2011), clustering (Karaboga and Ozturk, 2011), symbolic regression (Karaboga et al., 2012), image segmentation (Ma et al., 2011), constrained numerical optimization (Li and Yin, 2012), learning Bayesian networks (Ji et al., 2013).

Developed by Eusuff and Lansey in 2003 (Eusuff and Lansey, 2003), the shuffled frog-leaping algorithm (SFLA) is a member of the swarm intelligence family. It is a meta-heuristic optimization method inspired from the memetic evolution of frogs seeking food in a pond.This algorithm has been used in different applications such as Unit Commitment Problem (Ebrahimi et al., 2011), job shop scheduling (Li et al., 2012), economic dispatch (Niknam et al., 2011), clustering (Amiri et al., 2009), water resource management (Chung and Lansey, 2009), bandwidth scheduling (Xu et al., 2012).

In 2002, Li Xiao Lei proposed Artificial Fish Swarm Algorithm (AFSA) (Lei et al., 2002). It is based on the behavior of fish living in a swarm. This algorithm is used in different applications such as neural networks (Tsai and Lin, 2011), (Shen et al., 2011), color quantization (Yazdani et al., 2011), dynamic optimization problems (Yazdani et al., 2012), physics (Zheng and Lin, 2012), global optimization (Rocha et al., 2011), (Yazdani et al., 2011), (Yazdani et al., 2010), clustering (Zhu et al., 2012) (Yazdani et al., 2013), (Yazdani et al., 2010),

Harmony search (HS) is a meta-heuristic algorithm developed by Geem et al. in 2001 (Pan et al., 2010), which is inspired from the natural musical performance process that occurs when a musician searches for a better state of harmony. This algorithm has been used in various applications such as continuous optimization problems (Pan et al., 2010), knapsack (Zou et al., 2011), flow shop scheduling (Pan et al., 2011), damage detection (Miguel et al., 2012), economic dispatch (Pandi and Panihrahi, 2011), document clustering (Forsati et al, 2012), electronic (Wang et al., 2013).

Imperialistic cognitive Algorithm (ICA) was proposed in 2007 by Atashpaz and Lucas (Atashpaz-Gargari and Lucas, 2007). This algorithm is based on the social and political relations of imperialist and colonial countries. It has been used in applications such as structures (Kaveh and Talatahari, 2010), economic power dispatch (Mohammadi-ivatloo et al., 2012), template matching (Duan et al., 2010), electromagnetic (Coelho et al., 2012), electronics (Rashtchi et al., 2012).

Gravitational Search Algorithm (GSA) was proposed by Rashedi et al. in 2009. It is based on the physics laws related to the gravity of objects. It is used in various applications such as slope stability analysis (Khajehzadeh et al., 2012), power systems (Duman et al., 2012), image processing (Zhao, 2011), filter modeling (Rashedi et al., 2011), classification (Bahrololoum et al., 2012), discrete optimization (Rashedi et al., 2010), robot path planning (Purcaru et al., 2013).

3. FISH SWARM SEARCH ALGORITHM

In this section, a new algorithm is proposed for global optimization in continues stationary environments, which is called Fish Swarm Search Algorithm (FSSA). This algorithm is a population-based algorithm which is designed based on fish characteristics in a swarm.

In biology, an aggregation of fish is the general term for any collection of fish that have gathered in some locality. Fish aggregations can be structured or unstructured. An unstructured aggregation might be a group of mixed species and sizes that have been gathered randomly near some local resource, such as food or nesting sites.

If the aggregation is gathered in an interactive social way, they are said to be shoaling. Although shoaling fish can be related to each other in a loose way, where each fish swim and forage somewhat independently, they are nonetheless aware of the other members of the group as shown by the way they adjust behavior such as swimming, so as to remain close to the other fish in the group. Shoaling groups can include fish of disparate sizes and can include mixed-species subgroups.

Swarming behavior is defined as moving of fish in groups. The aim of moving in a group is to efficient food searching, since the group can perform better search, compared to individual ones. Swarming fish are usually of the same species and the same age or size. Fish swarms move with the individual members precisely spaced from each other. The swarms undertake complicated maneuvers, as though they have minds of their own. Many hypotheses to explain the function of swarming have been suggested so far e.g. better orientation, synchronized hunting, predator confusion, finding more food, and reduced risk of being found.

FSSA is based on finding more food and keeping the swarm according to its rules. In FSSA, the fish move toward better positions based on the hypothesis that better positions in the problem space contain more food. The fish move toward these better positions without losing their integrity. At first, the fish population is randomly distributed in the problem space with a uniform distribution. Then these fish will follow a procedure which will be described in following.

In general, the fish will follow those other fish which have found more food. So that they will find food and also help each other in finding more food. In FSSA, this behavior is simulated as follows: each fish i (F_i) chooses one fish in a better position randomly. Fish F_i, which resides at X_i, chooses fish F_j ($f(X_j) < f(X_i)$) in order to help it to find food. Using the Eq. (1) given below, this fish tries to approach its position:

$$Y_{i,d} = X_{i,d} + \left(\left(X_{j,d} - X_{i,d}\right) \times rand(0,2)\right) \tag{1}$$

in which $X_{i,d}$ is equal to the dthcomponent of the position vector of fish F_i, and rand(L,R) generates a random number with a uniform distribution in the range [L,R]. Y vector is a buffer vector. The schematic of movement of one fish toward a better one is illustrated in Fig. 1.

Every fish is shown with a circle in Fig. 1, and the bigger circle the fitter fish will be. In this figure, the fish F_i considers fitter fish in its swarm, and chooses one of them randomly with a uniform distribution. The next position of this fish, if better, will be a point on the 2D cubic feasible destination space, where the center of this space is the chosen better fish due to choosing a random number in the range [0,2].

After calculating the position of Y, first of all its components are checked to find that if it is placed in the search space. If any component is out of the search space, it will be set on the space's boundaries. After limiting the components of Y, its fitness value will be evaluated and if $f(Y_i)<f(X_i)$, the fish F_i will move to the position of Y_i, otherwise the fish F_i will start an individual search.

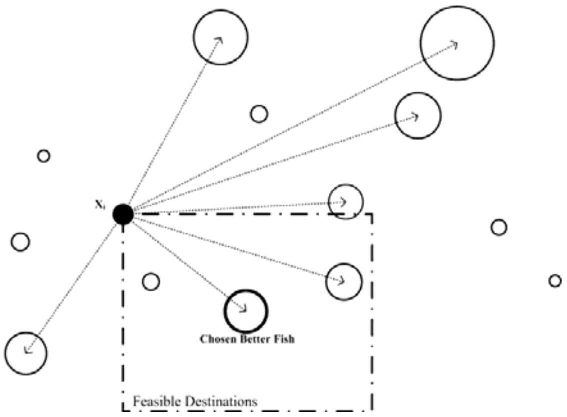


Figure 1: The schematic of following a better fish.

In the nature, fish can find their food using their sensors. These sensors which include sound, electrical, visual, and other sensors are different among fish. In FSSA, the comprehensible range of a fish is called CR. A fish can find food in its CR. In fact, a fish can attack its preys *rush_time* times before it gets tired. In FSSA, this behavior is simulated as follows. Suppose fish F_i wants to search individually for food. To do so, it will consider a random position T in its CR using:

$$T_d = X_{i,d} + \left(rand(-1,1) \times CR_{i,d}\right) \tag{2}$$

in which $CR_{i,d}$ is the d^{th}component of the vector CR for fish F_i. The value of CR vector for each fish in every iteration is obtained by using:

$$CR_{i,d} = \left(\left|X_{best,d} - X_{i,d}\right|\right) \tag{3}$$

in which X_{best} is the position of the best fish according to its fitness. After finding the T position by Eq.(2), and checking the search space boundary, its fitness value will be calculated and if $f(T)<f(X_i)$, the fish F_i will move to position T, otherwise its position will not be changed. A fish will try *rush_time* times to find its preys, which means it will execute Eq.(2) *rush_time* times according to its current position. The value of *rush_time* for each fish is obtained at the beginning of each iteration of the algorithm execution by using:

$$Rush\_time_i(t) = \left\lceil \log_2 (N - rank_i + 1) \right\rceil + 1 \tag{4}$$

in which N is the size of the population, and $rank_i$ is the rank of fish F_i according to its fitness value (the best fish in the rank one) in the swarm. Using Eq.(4), *rush_time* value for each fish is determined by its position corresponding to the position of the swarm members. In fact, those fish with better positions will be nearer to their prey, which means they will have more opportunity to hunt them. On the other hand, those fish with a worse position will be farther from their prey, giving them less opportunity to hunt their prey. Eq.(4) simulates the same idea, i.e. the value of *rush_time* for the fish with better positions is more than the value of *rush_time* for fish with the worse positions.

In Fig. 2, the schematic of searching for food is illustrated in a 2D space by fish F_i using Eq.(2). As it can be observed, the obtained positions in Eq.(2) are located in a D-dimensional cubic space with the center X_i. At the beginning of performing the behavior, the position of the best fish(X_{best}) is located at the corner of this space using Eq.(3) for obtaining CR. Suppose that two positions T_1 and T_2, obtained by Eq.(2) in this space, have a worse position than X_i. Therefore fish F_i will not move toward them. Then another position T_3 is calculated by Eq.(2), which shows a better fitness than X_i, so fish Fi changes its position to T_3. Then, fish F_i continues to search from its new position. Assume that this fish may find a better position in its fifth try. This procedure will continue *rush_time* times. Therefore, a fish may improve its position *rush_time* times at best, and its position may never improve at worst. If latter is the case, the fish will relocate in the swarm.

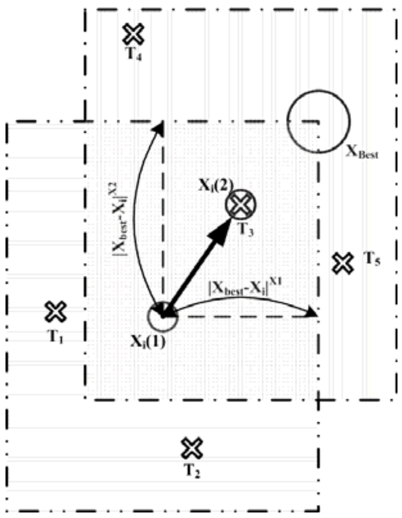


Figure 2: Schematic of individual search using Eq. (2).

The fish try to stay in the swarm in order to protect themselves from hunters and to find more food. They relocate in the swarm for two following reasons: 1) going toward the positions where they can find more food, and 2) to confuse the hunters and reduce the risk of being found. In this case, the fish try to move toward the center or toward the outside of the swarm, which leads to decreasing or increasing the size of the boundary surrounding the swarm. In FSSA, a behavior is also considered for fish relocation in the swarm. This behavior increases the fish opportunity for finding better positions, increases the diversity in the swarm, and increases the ability to escape from local optimums. At the beginning of each iteration, the central position of the swarm is calculated using:

$$X_{Center,d} = \frac{1}{N}\sum_{i=1}^{N} X_{i,d} \tag{5}$$

in which N is the population size. Suppose fish F_i is unsuccessful in its individual search, therefore it relocates itself using:

$$X_{i,d} = X_{i,d} + \left(\left(X_{Center,d} - X_{i,d}\right) \times rand(-1,1)\right) \tag{6}$$

After fish F_i executes Eq.(6), search space boundaries will be checked for its position. Fig. 3 illustrates the schematics of fish movement in a 2D space using Eq.(6). In Fig. 3, it is supposed that position *Star* is obtained using Eq.(5), and fish F_i executes Eq.(6) after failure in improving its position using Eq.(1) and (2), yielding its position on a point in the D-dimensional cubic feasible destination space. The center of this space will be the position of fish F_i, due to choosing random numbers in the range [-1,1]. The pseudo-code for FSSA is shown in Fig. 4.

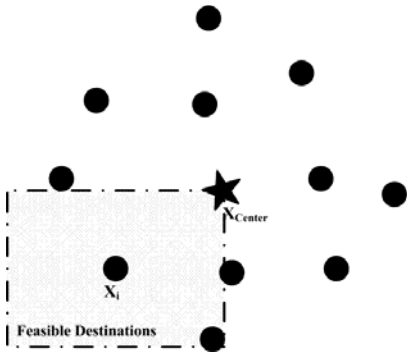


Figure 3: The relocation of Fish F_i using Eq.(6)

3.1. A note on Gbest in FSSA

In FSSA, the best fish cannot improve its position using Eq.(1) and (2), because it cannot find a better fish and its CR value is a zero vector using Eq.(3). This fish also does not need to execute Eq.(6), because relocation in this behavior is not due to improving in position, and it is only done for increasing diversity in the swarm and creating new positions. Therefore, the best fish in the swarm will not perform any moves. As a result, the position of the best fish in the swarm at the time *t*, is always the best position found by the algorithm by the time *t*.

3.2. Comparison with Artificial Fish Swarm Algorithm

AFSA, like FSSA, is originated from the fish swarm behavior in the nature but there are several significant differences between these two algorithms. In the following, the differences between AFSA and FSSA are explained.

In AFSA, there are four behaviors: *follow*, *prey* (or searching food), *swarm*, and *free move*. The fish search the problem space using these behaviors. In AFSA, the fish perform a *prey* behavior in order to perform a local search in their *visual* space. The basis for the *prey* behavior in AFSA is the parameter *visual*, which is a numeric parameter and is common between all fish. In standard AFSA, *visual* value is invariable during runtime, and in some enhanced versions, its value decreases during the runtime (Tsai and Lin, 2011) (Yazdani et al., 2011) (Yazdani et al., 2012). On the other hand, the basis of individual search in FSSA is the CR vector in Eq.(2), which is determined separately for every fish according to their vector distance with the best fish in the swarm. In AFSA, the fish may change their position *try_number* times, and this parameter is equal for all fish. However, in FSSA, each fish has its own *rush_time* value determined according to its position in the swarm, and may improve its position *rush_number* times.

```
FSSA:
1:  initialize $N$(population size)
2:  For each $Fish_i$ do
3:      initialize $X_i$ randomly in $D$-dimensional search space
4:  Endfor
5:  evaluate all Fish and determine $X_{best}$ (position of best fish)
6:  Repeat
7:      For each $Fish_i$ do
8:          obtain $CR_i$ by Eq.(3)
9:          obtain $rush\_time_i$ by Eq. (4)
10:     Endfor
11:     compute Center position $X_{center}$ by Eq. (5)
12:     For each $Fish_i$ do
13:         determine $Fish_j$ randomly which $f(X_j)<f(X_i)$
14:         obtain $Y_i$ by Eq. (1)
15:         check search space boundary for $Y_i$
16:         If $f(Y_i)<f(X_i)$ then
17:             $X_i=Y_i$
18:         Else
19:             $Flag_i=0$;
20:             For $counter=1$ to $rush\_time_i$ do
21:                 obtain $T$ by Eq. (2)
22:                 check search space boundary for $T$
23:                 If $f(T)<f(X_i)$ then
24:                     $X_i=T$
25:                     $Flag_i=1$;
26:                 Endif
27:             Endfor
28:             If $Flag_i==0$ then
29:                 obtain $X_i$ by Eq. (6)
30:                 check search space boundary for $X_i$
31:             Endif
32:         Endif
33:     Endfor
34: Until stopping criterion is met
```

Figure 4: Pseudo-code for FSSA.

In AFSA, a fish may move to a random position in its visual when it cannot improve its position, but in FSSA and under the same circumstances, the behavior of the next position of the fish is determined based on Eq.(6). In AFSA, the *swarm* behavior is performed according to the *central position*, which is the same as Eq. (6) in FSSA. But in this behavior, all fish move one step toward the *central position* according to parameter *step*, after assessing conditions such as superiority of the central position compared to the positions of the fish. In fact, the behavior of the swarm is performed with the aim of improving the position, while Eq.(6) is executed with the aim of increasing diversity in the swarm. *Follow* behavior in AFSA means every fish searches for better neighbors in its *visual* space. Finding neighbors in AFSA indicates a high computational load for the calculation of Euclidean distances between all pairs of fish. In FSSA, however, fish follow one of the better fish in the swarm regardless of their positions.

In AFSA, the best found position by the swarm is stored in bulletin, and any fish (even the best fish) may move toward the worse positions. In fact, the best found position by AFSA cannot be used for the purpose of improving the optimization process. On the other hand, in FSSA, the best found position by the swarm is the same as the position of the best fish. Overall, in AFSA there are several parameters such as step, crowd factor, visual, and try_number (Tsai and Lin, 2011), where their setting in order to perform efficient global and local searches is a complex task. Thus, these parameters are not involved in FSSA.

The movement equations in AFSA and FSSA have significant differences with each other. FSSA execution steps are very simple, as shown in Fig. 4. On the contrary, the behaviors swarm and follow in AFSA are executed simultaneously, and if any fails, the prey behavior will be executed, and if prey fails, free move behavior will be executed. After the execution of these two behaviors, the fish will accept the one with better results. In fact, the other path will have no effects on the algorithm progress which results in a considerable increase in computational load.

4. EXPERIMENTAL SETUP AND RESULTS

In this section, the experiments are performed on standard benchmark functions. These benchmark functions are widely utilized in benchmarking global optimization algorithms (Pan et al., 2010), (Yao et al., 1999), (Suganthan et al., 2005). In this paper, these functions are divided into three groups. The first group includes twelve unimodal functions (Table1). The second group includes eleven multimodal functions with various dimensionalities (Table 2). Finally, the third group includes two shifted multimodal functions (f_{24},f_{25}), one shifted unimodal function (f_{26}), two rotated multimodal functions (f_{27},f_{28}) and 2 shifted-rotated multimodal functions (f_{29},f_{30}) (Table 3) (Suganthan et al., 2005). The simulations are performed using MATLAB Ver. 7.8.0.347 on an Intel Core i7 860 with 8GB of RAM and with an operating system of Win7 Ultimate 64bits.

Table 1: Unimodal Benchmark Functions

Name	Test Function	D	Search range	f_{Min}	Accept
Matyas	$f_1(x)=0.26\left(x_1^2+x_2^2\right)-0.48x_1x_2$	2	$[-10,10]^D$	0	0
Easom	$f_2(x)=-\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)$	2	$[-100,100]^D$	-1	-0.99
Noise	$f_3(x)=\sum_{i=1}^{D} ix_i^4 + random\ [0,1)$	30	$[-1.128,1.128]^D$	0	0.01
Zakharov	$f_4(x)=\sum_{i=1}^{D} x_i^2+\left(\sum_{i=1}^{D} 0.5ix_i\right)^2+\left(\sum_{i=1}^{D} 0.5ix_i\right)^4$	10	$[-5,10]^D$	0	0.01
Trid10	$f_5(x)=\sum_{i=1}^{D}(x_i-1)^2-\sum_{i=2}^{D} x_ix_{i-1}$	10	$[-100,100]^D$	-210	-209.99
Schwefel 2.22	$f_6(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	30	$[-10,10]^D$	0	0.01
Step	$f_7(x)=\sum_{i=1}^{D}\left(\lfloor x_i+0.5\rfloor\right)^2$	30	$[-100,100]^D$	0	0
Hyper-ellipsoid	$f_8(x)=\sum_{i=1}^{D} ix_i^2$	30	$[-5.12,5.12]^D$	0	0.01
Sum of different power	$f_9(x)=\sum_{i=1}^{D}\lvert x_i\rvert^{i+1}$	30	$[-1,1]^D$	0	0.01
Schwefel 1.2	$f_{10}(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$	30	$[-65.536,65.536]^D$	0	10
Sphere	$f_{11}(x)=\sum_{i=1}^{D} x_i^2$	30	$[-100,100]^D$	0	0.01
Schwefel 2.21	$f_{12}(x)=\max\left(\lvert x_i\rvert, 1\le i\le D\right)$	30	$[-100,100]^D$	0	0.01

Table 2: Multimodal Benchmark Functions

Name	Test Function	D	Search range	f_{Min}	Accept
Bohachevsky1	$f_{13}(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7$	2	$[-100,100]^D$	0	0
Bohachevsky2	$f_{14}(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)(4\pi x_2)+0.3$	2	$[-100,100]^D$	0	0
Bohachevsky3	$f_{15}(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1+4\pi x_2)+0.3$	2	$[-100,100]^D$	0	0
Schaffer	$f_{16}(x)=0.5+\frac{\sin^2\left(\sqrt{x_1^2+x_2^2}\right)-0.5}{\left(1+0.001\left(x_1^2+x_2^2\right)\right)^2}$	2	$[-100,100]^D$	0	0.01
Butterfly	$f_{17}(x)=\left(x_1^2-x_2^2\right)\sin(x_1+x_2)/\left(x_1^2+x_2^2\right)$	2	$[-10,10]^D$	-1	-0.99
Six Hump Camel Back	$f_{18}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	10	$[-5,5]^D$	-1.03163	-1.03
Ackley	$f_{19}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^D x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^D\cos 2\pi x_i\right)+20+e$	30	$[-32,32]^D$	0	0.01
Weierstrass	$f_{20}(x)=\sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}\left(a^k\cos\left(2\pi b^k(x_i+0.5)\right)\right)\right)-D\sum_{k=0}^{k\max}\left(a^k\cos\left(2\pi b^k\times 0.5\right)\right)$ $a=0.5, b=3, k_{\max}=20$	30	$[-0.5,0.5]^D$	0	0.01
Griwank	$f_{21}(x)=\frac{1}{4000}\sum_{i=1}^D x_i^2-\prod_{i=1}^D\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$[-600,600]^D$	0	0.01
Penalized1	$f_{22}(x)=\frac{\pi}{D}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2\left(1+10\sin^2(\pi y_{i+1})\right)+(y_D-1)^2\right\}$ $+\sum_{i=1}^D u(x_i,10,100,4)$ $y_i=1+\frac{1}{4}(x_i+1)$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	30	$[-50,50]^D$	0	0.01
Penalized2	$f_{23}(x)=0.1\left\{\sin^2(\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2\left(1+\sin^2(3\pi x_{i+1})\right)+(x_D-1)^2\left(1+\sin^2(2\pi x_D)\right)\right\}$ $+\sum_{i=1}^D u(x_i,5,100,4)$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	30	$[-50,50]^D$	0	0.01

Table 3: Shifted and Rotated Benchmark Functions

Name	Test Function	D	Search range	f_{Min}	Accept
Shifted Ackly	$f_{24}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^D z_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^D\cos 2\pi z_i\right)+20+e$ $z=x-o^{**}\quad bias_{24}=-140$	30	$[-32,32]^D$	-140	-139.99
Shifted Griwank	$f_{25}(x)=\frac{1}{4000}\sum_{i=1}^D z_i^2-\prod_{i=1}^D\cos\left(\frac{z_i}{\sqrt{i}}\right)+1+bias_{25},$ $z=x-o^{**}, bias_{25}=-180$	30	$[-600,600]^D$	-180	-179.99
Shifted Sphere	$f_{26}(x)=\sum_{i=1}^D z_i^2+bias_{sphere}\quad z=x-o^{**},\ bias_{26}=-450$	30	$[-100,100]^D$	-450	-449.99

Rotated Penalized1	$f_{27}(x)=\frac{\pi}{D}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(w_i-1)^2\left(1+10\sin^2(\pi y_{i+1})\right)+(y_D-1)^2\right\}+\sum_{i=1}^{D}u(x_i,10,100,4)$ $y_i=1+\frac{1}{4}(w_i+1)$ $u(w_i,a,k,m)=\begin{cases}k(w_i-a)^m, & w_i>a\\ 0, & -a\le w_i\le a\\ k(-w_i-a)^m, & w_i<-a\end{cases}$ $w=x\times M^*$	30	$[-50,50]^D$	0	0.01
Rotated Penalized2	$f_{28}(x)=0.1\left\{\sin^2(\pi w_1)+\sum_{i=1}^{D-1}\begin{matrix}(w_i-1)^2\left(1+\sin^2(3\pi w_{i+1})\right)\\ +(w_D-1)^2\left(1+\sin^2(2\pi w_D)\right)\end{matrix}\right\}+\sum_{i=1}^{D}u(w_i,5,100,4)$ $u(w_i,a,k,m)=\begin{cases}k(w_i-a)^m, & w_i>a\\ 0, & -a\le w_i\le a\\ k(-w_i-a)^m, & w_i<-a\end{cases}$ $w=x\times M^*$	30	$[-50,50]^D$	0	0.01
Shifted Rotated Ackly	$f_{29}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}z_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi z_i\right)+20+e+bias_{29}$ $z=(x-o^{**})\times M^*\quad bias_{29}=-140$	30	$[-32,32]^D$	-140	-139.99
Shifted Rotated Griwank	$f_{30}(x)=\frac{1}{4000}\sum_{i=1}^{D}z_i^2-\prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right)+1+bias_{30}$, $z=(x-o^{**})\times M^*,\ bias_{30}=-180$	30	$[-600,600]^D$	-180	-179.99

4.1. Effects of Parameter Setting

One of the major disadvantages of the meta-heuristic algorithms is that they require considerable knowledge concerning the value of the algorithm parameter. If the algorithm includes more complicated structures consisting of additional parameters, the degrees of freedom correspondingly decrease.

FSSA is designed so that there is a minimum parameter setting requirement, with population size being the only parameter which needs to be initialized. The execution terminating state is 200,000 fitness evaluations, and the experiments have been performed 100 times with different random seeds for each experiment. The average of obtained results from the proposed algorithm in 100 executions and its standard deviation with different population sizes on thirty benchmark functions has been illustrated in Table 4. As it can be seen on Table 4, the algorithm performance is not satisfactory when the population size is 10. In general, the best results occur when the population size is 20 in optimization of unimodal functions. However, the obtained results in multimodal and shifted and rotated functions are not satisfactory with a population size of 20. As the population size increases, the performance of the proposed result deteriorates in most of the unimodal functions, but it improves in multimodal and shifted and rotated functions. The reason is that the proposed algorithm terminating state is reaching the maximum number of the fitness evaluation. In unimodal functions with smaller population sizes, the algorithm may execute more times before reaching the terminating state, and performs exploitation in a better way. However, the small population size may result in premature convergence, which decreases the performance on multimodal functions. On the other hand, increasing the population size enhances the exploration ability, but the algorithm is executed fewer times before the terminating state which leads to a decreased ability in exploitation. As a result, the performance deteriorates in simple unimodal functions and enhances in more complex functions.

Table 4: The average and standard deviation obtained from 100 times executing FSSA on the benchmark functions mentioned in Tables 1 to 3**.**

F	Population Size						
	10	20	30	40	50	75	100
f_1	0.0001 (0.0004)	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_2	**-1 (0)**	**-1 (0)**	**-1 (0)**	**-1 (0)**	**-1 (0)**	**-1 (0)**	**-1 (0)**
f_3	0.2622 (0.7591)	0.0042 (0.0014)	0.0050 (0.0016)	0.0044 (0.0016)	**0.0041 (0.0011)**	0.0042 (0.0014)	0.0043 (0.0012)
f_4	0.3672 (0.8729)	**1.39e-184 (0)**	1.75e-142 (6.18e-142)	8.78e-110 (3.93e-109)	**1.29e-96 (2.29e-96)**	5.89e-65 (2.11e-64)	5.04e-50 (1.40e-49)
f_5	-8.8742 (214.8832)	-209.9999 (8.06e-13)	-209.9999 (5.80e-13)	-209.9999 (7.86e-13)	**-210.00 (6.89e-13)**	-209.9999 (1.60e-12)	-209.9999 (2.16e-11)
f_6	3.2809 (5.6334)	**1.47e-99 (2.12e-99)**	2.60e-67 (4.23e-67)	1.15e-49 (1.40e-49)	2.96e-39 (3.88e-39)	1.90e-25 (2.55e-25)	7.35e-19 (5.86e-19)
f_7	500 (2236.0679)	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_8	5.6580 (23.3734)	**2.20e-143 (5.82e-143)**	3.26e-103 (4.56e-103)	1.0090e-75 (3.10e-75)	1.41e-62 (1.55e-62)	3.05e-41 (5.05e-41)	3.17e-30 (4.54e-30)
f_9	5.77e-07 (1.94e-06)	**1.67e-230 (0)**	8.90e-188 (0)	4.74e-157 (1.94e-156)	6.8e-126 (6.5e-125)	2.92e-90 (9.08e-90)	3.25e-70 (1.32e-69)
f_{10}	782.5799 (1325.9946)	**6.58e-10 (1.13e-09)**	1.18e-06 (1.82e-06)	7.71e-05 (0.0001)	0.0022 (0.0028)	0.1703 (0.1667)	2.1806 (1.5447)
f_{11}	17.5221 (35.9711)	**7.19e-142 (1.21e-141)**	2.17e-100 (7.78e-100)	2.32e-75 (3.64e-75)	1.40e-60 (5.96e-60)	3.27e-39 (7.12e-39)	2.18e-28 (3.13e-28)
f_{12}	60.0268 (12.6718)	1.36e-05 (2.56e-05)	2.43e-07 (3.87e-07)	**1.91e-07 (2.58e-07)**	6.62e-07 (3.08e-07)	4.55e-05 (4.21e-05)	0.0004 (0.0002)
f_{13}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{14}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{15}	3.30e-05 (9.45e-05)	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{16}	0.0131 (0.0153)	0.0072 (0.0043)	0.0068 (0.0045)	0.0063 (0.0047)	0.0058 (0.0041)	0.0052 (0.0048)	**0.0034 (0.0047)**
f_{17}	-0.9953 (0.0090)	-0.9980 (0.0046)	-0.9969 (0.0067)	-0.9990 (0.0042)	**-1 (0)**	**-1 (0)**	**-1 (0)**
f_{18}	**-1.0316 (2.22e-16)**	**-1.0316 (2.27e-16)**	**-1.0316 (2.27e-16)**	**-1.0316 (2.27e-16)**	**-1.0316 (2.24e-16)**	**-1.0316 (2.27e-16)**	**-1.0316 (2.27e-16)**
f_{19}	11.7642 (4.5559)	0.6403 (0.7639)	0.0577 (0.2582)	**2.66e-15 (0)**	**2.66e-15 (0)**	**2.66e-15 (0)**	4.26e-15 (2.69e-15)
f_{20}	11.2779 (7.3006)	0.4331 (0.5984)	0.0247 (0.0594)	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{21}	7.3211 (20.7217)	0.0228 (0.0174)	0.0426 (0.0368)	0.0266 (0.0258)	**0.0136 (0.0118)**	0.0154 (0.0192)	0.0178 (0.0188)
f_{22}	7.2889 (4.5264)	0.3019 (0.7658)	0.1663 (0.3845)	0.0622 (0.1628)	**1.57e-32 (3.57e-47)**	0.0155 (0.0379)	0.0362 (0.0608)
f_{23}	53530.30 (236635.8)	0.0101 (0.0312)	0.0090 (0.0278)	7.86e-32 (3.32e-31)	1.67e-33 (4.17e-34)	**1.66e-33 (7.54e-34)**	4.91e-28 (1.07e-27)
f_{24}	-125.490 (2.8897)	-138.3845 (2.3822)	**-140.0000 (0)**	**-140.0000 (0)**	**-140.0000 (0)**	-140.0000 (1.59e-14)	-140.0000 (2.52e-14)
f_{25}	-172.7130 (16.8704)	-179.8581 (0.2240)	-179.9748 (0.0214)	-179.9837 (0.0163)	**-179.9999 (0.0156)**	-179.9829 (0.0001)	-179.9799 (0.0212)
f_{26}	1348.2567 (3221.755)	-445.4439 (14.5035)	**-450 (0)**	**-450 (0)**	**-450 (0)**	**-450 (0)**	**-450 (0)**
f_{27}	19.1294 (4.5643)	2.9363 (5.9199)	0.1583 (0.5648)	0.0362 (0.0967)	**4.31e-29 (6.19e-29)**	1.21e-20 (3.23e-20)	0.0103 (0.03198)
f_{28}	1.80e+06 (7.01e+06)	0.0090 (0.0278)	3.09e-10 (1.516e-09)	1.04e-09 (2.12e-09)	**2.37e-11 (3.63e-11)**	6.11e-09 (8.29e-09)	1.70e-08 (1.84e-08)
f_{29}	-128.03 (3.9019)	-137.5503 (2.2893)	-139.7297 (0.5713)	-139.7131 (0.5365)	**-140.0000 (0)**	-139.9534 (0.2082)	**-140.0000 (0)**
f_{30}	-155.29 (53.3888)	-179.9114 (0.2409)	-179.9713 (0.0248)	-179.9833 (0.0187)	-179.9871 (0.0046)	-179.9840 (0.0153)	**-179.9858 (0.0133)**

4.2. Comparison with Other Algorithms

In this section the performance of the proposed algorithm is compared to those of nine other state-of-the-art algorithms including Modified Artificial fish swarm algorithm with communication behavior(CM-AFSA) (Tsai and Lin, 2011), global-best particle swarm optimization(GPSO) (Shi and Eberhart, 1998), Local-best particle swarm optimization(L-PSO) (Kennedy and Mondes, 2002), Adaptive inertia weight particle swarm optimization(AIW-PSO) (Nickabadi et al., 2011), Global-best harmony search (Omran and Mahdavi, 2007), Improved imperialist cognitive algorithm(I-ICA) (Shirkouhi et al., 2010), gravitational search algorithm(GSA) (Rashedi et al., 2009), modified artificial bee colony algorithm(M-ABC) (Akay and Karaboga, 2012) and shuffled frog leaping algorithm(SFLA) (Eusuff and Lansey, 2003). These algorithms have been selected based on their high performance, their vast application, and their relevance to the topic. Absorbing wall method (Huang and Mohan, 2005), (Robinson and Rahmat-samii, 2004) has been used to keep the particles in the problem space to implement PSO algorithms. The experiments in this section have been performed on 30 benchmark functions of Tables 1 to 3. It is worth to mention that the comparative algorithms are implemented exactly based on the references.

The experiments have been done 100 times and with different random seeds. The values of the algorithm's parameters have been shown in Table 5. These values have been determined by the relevant references of each algorithm. It must be noted that since every algorithm has its own structure and in every iteration of its execution, different fitness evaluation numbers are performed, the execution terminating state for all algorithms is defined as reaching 200,000 fitness evaluations. This way, those algorithms with a higher number of fitness evaluations in every iteration will be executed a smaller number of iterations, and those algorithms with a smaller number of fitness evaluations in every iteration will execute a higher number of iterations. The population size is equal to 50 for the proposed algorithm, because as shown in the obtained results in Table 4, the performance of the proposed algorithm in all benchmark functions is satisfactory with this population size. It could be concluded in the course of experiments that the performance of other algorithms decreases in many cases with a population size of 50 in 200,000 fitness evaluations. For instance, with a population size of 20, PSO algorithms should be executed around 10,000 iterations in order to reach the terminating state of reaching 200,000 fitness evaluations. The obtained results in this case are generally more satisfactory than when PSO execute 4,000 iterations in order to reach 200,000 fitness evaluations with a population size of 50.

Choosing a maximum number of execution iterations does not provide fair conditions for comparison of algorithms. The reason is that these algorithms deal with different procedures in every iteration. For example, PSO algorithm performs as many fitness evaluations as its population size in every iteration. But ABC or CM-AFSA algorithms perform more fitness evaluations than their population sizes, i.e. an iteration of execution in every algorithm executes different fitness evaluations. In fact, reaching a maximum number of fitness evaluations as a terminating state is reasonable, but reaching a maximum number of iterations is not a fair criterion, even if algorithm's populations are set as equal. The average of the obtained results from algorithms along with their standard deviation is illustrated in Table 6. For every benchmark function, the best obtained result is bolded.

As shown in the Table 6, for some benchmark functions such as benchmark functions 1, 2, 7, 13, 14, 15, 17, and 18, several algorithms have been able to reach best results. For unimodal functions, the proposed algorithm has obtained best results in 9 out of 12 cases in Table 1, and has reached satisfactory results in the other 3 cases. On the other hand, for benchmark function 11 so called *Sphere*, the proposed algorithm has obtained better results than AIW-PSO with a population size of

20. The obtained results from the proposed algorithm for unimodal functions imply the high ability of the algorithm in performing exploitation.

Table 5: Setting Algorithms Parameters

Algorithm	Parameter Setting
FSSA	Population size = 50
G-PSO	Population size =20 c_1=c_2=2 W_{min}=0.4 W_{max}=0.9 neighbor topology=Global star Inertia weight = linear decrement
L-PSO	Population size =20 c_1=c_2=2 W_{min}=0.4 W_{max}=0.9 neighbor topology=ring Inertia weight = linear decrement
AIW-PSO	Population size =20 c_1=c_2=2 W_{min}=0 W_{max}=1 neighbor topology=Global star Inertia weight = Adaptive[4]
G-HS	Population size(HMS) =5 HMCR=0.9 PAR_{min}=0.01 PAR_{max}=0.99

CM-AFSA	Population size =50
	try-number=5
	crowd factor=0.75
	Visual=Par[1]
	Step = Visual/5
	c_3=c_6=2

GSA	Population size =50 α=20 G_0=100 k_0=50
I-ICA	Country number=200 empire number = 10 revolution rate=0.3 ξ=0.1 β=2
M-ABC	Population size =40 Modification rate=0.8 Limit = 0.5 ×Dimension ×Population size Scaling factor = 1
SFLA	Population size = 30 number of memeplex=10 D_{max}=0.2 * Function bound Iteration in memeplex = 10

In multimodal functions, the performance of the proposed algorithm is still higher than that of other algorithms. In multimodal functions, the obtained results from the proposed algorithm are worse than other algorithms only in two cases, and better than other algorithms in eight cases. After several experiments, it was concluded that the proposed algorithm will find the optimum position of benchmark function 16 in all executions with a population size of 500.

Table 6: Comparing the average and standard deviation of the obtained results from 10 algorithms on functions in Tables 1 to 3

FUNCTION	Algorithms									
	G-PSO	L-PSO	G-HS	I-ICA	GSA	M-ABC	CM-AFSA	SFLA	AIW-PSO	FSSA
f_1	**0 (0)**	**0 (0)**	6.25e-05 (4.36e-05)	**0 (0)**	2.02e-22 (1.99e-22)	1.66e-20 (4.81e-20)	3.41e-26 (6.47e-26)	1.54e-28 (1.05e-27)	**0 (0)**	**0 (0)**
f_2	**-1** (0)	**-1** (0)	-0.78 (0.40)	**-1 (0)**	-0.8169 (0.3238)	**-1 (0)**	**-1 (0)**	**-1 (0)**	**-1 (0)**	**-1 (0)**
f_3	0.79 (1.77)	0.02 (0.008)	**0.002 (0.001)**	0.26 (0.11)	0.01 (0.0040)	0.01 (0.006)	0.01 (0.0041)	**0.002 (0.0006)**	0.262 (0.783)	0.004 (0.001)
f_4	19.07 (37.72)	2.85 (8.67)	0.0001 (0.0003)	3.22e-61 (1.25e-60)	3.14e-18 (1.19e-18)	1.14e-15 (1.59e-15)	3.87e-14 (1.18e-13)	1.22e-09 (2.83e-09)	19.423 (33.128)	**1.29e-96 (2.29e-96)**
f_5	-148.03 (438.17)	-209.99 (0.0001)	-198.32 (8.16)	-209.99 (3.25e-06)	**-210.00 (4.60e-13)**	-209.98 (0.01)	-209.99 (0.0001)	-208.94 (0.3936)	-209.99 (1.69e-07)	**-210.00 (6.89e-13)**
f_6	2105.73 (4441.42)	333.60 (190.37)	0.01 (0.01)	4.34-e10 (1.55e-07)	1.81e-08 (2.09e-09)	1.43e-08 (5.07e-09)	0.0009 (0.0008)	5.20e-11 (2.59e-09)	408.56 (1246.75)	**2.96e-39 (3.88e-39)**
f_7	**0 (0)**	**0 (0)**	**0 (0)**	5.90 (4.20)	**0 (0)**	**0 (0)**	0.7400 (0.8762)	**0 (0)**	0.04 (0.197)	**0 (0)**
f_8	102.76 (82.19)	3.14 (13.64)	3.43e-05 (6.72e-05)	1.11e-20 (6.13e-20)	1.11e-16 (2.99e-17)	3.68e-11 (2.11e-11)	5.08e-08 (8.15e-08)	3.14e-13 (2.10e-12)	77.07 (90.72)	**1.41e-62 (1.55e-62)**
f_9	1.24e-85 (6.08e-85)	3.64e-45 (1.52e-44)	5.29e-08 (8.84e-08)	1.18e-25 (2.21e-25)	1.01e-43 (6.98e-43)	2.78e-14 (4.99e-14)	1.76e-10 (4.15e-10)	3.82e-14 (4.81e-14)	1.38e-75 (9.77e-075)	**6.8e-126 (6.5e-125)**
f_{10}	22927.18 (16684.59)	24607.91 (15669.75)	733.88 (1566.06)	2.92 (2.14)	1.61 (1.98)	4217.89 (3296.54)	17.20 (9.81)	1. 23 (0.63)	12647.72 (12696.02)	**0.0022 (0.0028)**
f_{11}	2.48e-50 (3.82e-51)	1.33e-20 (3.11e-20)	0.0009 (0.001)	1.27e-19 (3.06e-19)	1.29e-17 (2.61e-18)	1.35e-09 (6.55e-10)	7.19e-08 (4.57e-08)	2.48e-18 (5.10e-18)	**7.86e-127 (5.56e-126)**	1.40e-60 (5.96e-60)
f_{12}	1.63 (1.19)	9.84 (3.06)	6.40 (3.97)	37.06 (7.50)	**1.99e-09 (2.69e-10)**	3.43 (3.44)	3.21 (2.89)	3.85 (0.87)	5.088 (2.549)	6.62e-07 (3.08e-07)
f_{13}	**0 (0)**	**0 (0)**	0.0008 (0.001)	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{14}	**0 (0)**	**0 (0)**	3.32e-05 (5.00e-05)	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{15}	**0 (0)**	**0 (0)**	0.02 (0.02)	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{16}	**0 (0)**	**0 (0)**	0.01 (0.01)	0.003 (0.004)	0.01 (0.0036)	0.0001 (0.0001)	9.79e-5 (0.0004)	**0 (0)**	0.0009 (0.002)	0.006 (0.004)
f_{17}	-0.99 (1.219e-09)	-0.99 (6.37e-09)	-0.99 (0.0006)	**-1 (0)**	-0.99 (0.0006)	-0.99 (1.38e-05)	**-1 (0)**	**-1 (0)**	-0.999 (1.45e-09)	**-1 (0)**
f_{18}	**-1.0316 (2.24e-16)**	**-1.0316 (2.24e-16)**	**-1.0316 (5.79e-06)**	**-1.0316 (3.04e-16)**	**-1.0316 (2.24e-16)**	**-1.0316 (2.24e-16)**	**-1.0316 (2.30e-16)**	**-1.0316 (2.24e-16)**	**-1.0316 (1.56e-015)**	**-1.0316 (2.24e-16)**
f_{19}	0.98 (2.79)	4.72e-11 (1.24e-10)	0.005 (0.004)	13.28 (3.74)	2.77e-09 (2.90e-10)	1.63e-05 (5.30e-06)	1.49 (0.92)	1.05e-06 (4.36e-05)	2.5554 (2.0871)	**2.66e-15 (0)**
f_{20}	35.55 (3.33)	36.00 (2.30)	0.1455 (0.0837)	22.47 (3.51)	0.0002 (2.72e-05)	0.0005 (0.0001)	16.08 (3.98)	1.58 (0.92)	32.3931 (4.015)	**0 (0)**
f_{21}	0.05 (0.05)	0.01 (0.01)	**0.002 (0.008)**	0.10 (0.11)	0.06 (0.08)	**0.002 (0.006)**	0.07 (0.04)	0.05 (0.06)	0.032 (0.041)	0.014 (0.012)
f_{22}	0.09 (0.22)	1.63e-15 (9.93e-15)	5.36e-06 (6.44e-06)	1.37 (1.57)	0.002 (0.01)	0.002 (0.008)	2.07 (1.83)	7.03e-13 (4.65e-12)	0.3225 (0.5845)	**1.57e-32 (3.57e-47)**
f_{23}	2.56e-30 (1.12e-29)	1.80e-17 (4.84e-17)	1.59e-05 (2.67e-05)	0.03 (0.05)	3.70e-13 (1.69e-13)	0.0002 (0.0004)	0.03 (0.08)	2.80e-11 (1.01e-10)	0.0018 (0.0128)	**1.67e-33 (4.17e-34)**
f_{24}	-119.91 (1.88)	-119.48 (0.76)	-134.34 (0.55)	-122.14 (2.67)	-139.99 (3.57e-10)	-139.99 (1.31e-05)	-135.29 (6.22)	-121.46 (0.37)	-121.00 (3.633)	**-140.00 (0)**
f_{25}	-54.95 (100.17)	-130.95 (66.46)	-176.04 (0.93)	-179.90 (0.11)	-127.3782 (13.5240)	-179.998 (0.001)	-179.90 (0.07)	-155.18 (3.38)	-135.81 (42.06)	**-179.999 (0.02)**
f_{26}	22780.16 (11641.13)	25750.37 (13525.86)	-105.65 (94.76)	-449.99 (6.10e-13)	-450 (4.52e-14)	-449.99 (3.37e-10)	-449.99 (5.19e-08)	6079.01 (741.70)	15050.09 (11900.14)	**-450 (0)**
f_{27}	236.48 (2813.94)	3.26 (1.87)	0.73 (1.29)	18.06 (6.67)	1.10e-19 (3.12e-20)	0.09 (0.08)	4.26 (2.83)	2.84 (1.55)	1738.52 (6508.67)	**4.31e-29 (6.19e-29)**
f_{28}	174178.62 (839389.58)	0.004 (0.02)	86 (0.55)	52.09 (9.62)	2.05e-09 (1.67e-09)	8.96e-05 (0.0002)	0.01 (0.04)	12.30 (17.02)	153385.30 (761306.83)	**2.37e-11 (3.63e-11)**
f_{29}	-119.71 (1.59)	-119.38 (0.85)	-133.15 (0.85)	-122.58 (2.85)	-139.99 (3.32e-10)	-139.99 (0.0002)	-137.26 (4.44)	-121.56 (0.28)	-120.55 (2.819)	**-140.00 (0)**
f_{30}	-87.59 (78.02)	-127.12 (68.63)	-176.27 (0.80)	-179.91 (0.09)	-129.39 (12.77)	-179.60 (0.05)	-179.91 (0.06)	-154.42 (3.88)	-119.42 (56.98)	**-179.99 (0.005)**

The noticeable point in the proposed algorithm is that its performance has not deteriorated notably in shifted and rotated functions, unlike other algorithms. Among Table 3 functions, the obtained results from the proposed algorithm are the best in all cases. The obtained result from the proposed algorithm in Griwank function (F_{21}) is not the best. However, other algorithms encounter a decreased performance in shifted and rotated functions, and as a result, the proposed algorithm was able to obtain a better result in the shifted (F_{25}) and shifted and rotated (F_{30}) of Griwank function while maintaining its performance level, as compared with other algorithms.

To better understanding of the algorithms performance comparison, the algorithms ranks have been illustrated in Table 7 according to the average of their obtained results. The ranking is as follows: an algorithm rank minus one shows how many other algorithms have obtained better results than that. For example, when two algorithms are commonly ranked 1, the next algorithm is ranked 3, because two algorithms have obtained better results. As can be seen, the proposed algorithm has reached rank 1 in 25 cases. The sum of algorithms ranks are shown in the last row of Table 7. As it can be observed, the proposed algorithm has obtained better results than other algorithms in general.

Table 7: Algorithms ranks according to their average obtained results on 30 benchmark functions

F	Algorithms									
	G-PSO	L-PSO	G-HS	I-ICA	GSA	M-ABC	CM-AFSA	SFLA	AIW-PSO	FSSA
f_1	1	1	10	1	8	9	7	6	1	1
f_2	1	1	10	1	9	1	1	1	1	1
f_3	10	7	1	8	4	4	4	1	9	3
f_4	9	8	7	2	3	4	5	6	10	1
f_5	10	3	9	3	1	7	3	8	3	1
f_6	10	8	7	3	5	4	6	2	9	1
f_7	1	1	1	10	1	1	9	1	8	1
f_8	10	8	7	2	3	5	6	4	9	1
f_9	2	4	10	6	5	7	9	8	3	1
f_{10}	9	10	6	4	3	7	5	2	8	1
f_{11}	3	4	10	5	7	8	9	6	1	2
f_{12}	3	9	8	10	1	5	4	6	7	2
f_{13}	1	1	10	1	1	1	1	1	1	1
f_{14}	1	1	10	1	1	1	1	1	1	1
f_{15}	1	1	10	1	1	1	1	1	1	1
f_{16}	1	1	9	7	9	5	4	1	6	8
f_{17}	5	5	5	1	5	5	1	1	5	1
f_{18}	1	1	1	1	1	1	1	1	1	1
f_{19}	7	2	5	10	3	5	8	4	9	1
f_{20}	9	10	4	7	2	3	6	5	8	1
f_{21}	6	3	1	10	8	1	9	6	5	4
f_{22}	7	2	4	9	5	5	10	3	8	1
f_{23}	2	3	6	9	4	7	9	5	8	1
f_{24}	9	10	5	6	2	2	4	7	8	1
f_{25}	10	8	5	3	9	2	3	6	7	1
f_{26}	9	10	6	3	1	3	3	7	8	1
f_{27}	9	6	4	8	2	3	7	5	10	1
f_{28}	10	4	8	7	2	3	5	6	9	1
f_{29}	9	10	5	6	2	2	4	7	8	1
f_{30}	10	8	5	2	7	4	2	6	9	1
Total	*176*	*150*	*189*	*147*	*115*	*116*	*147*	*124*	*181*	***44***

In order to compare algorithms convergence speed and reliability, further experimental results are presented and compared in Tables 8-10. The results tabulated in Table 8 are the average fitness evaluations needed to reach the threshold presented as acceptable solutions specified in Tables 1-3. In addition, note that the average fitness evaluations are calculated only for the runs that have been “successful.” The word “successful” is used when an algorithm reaches the satisfactory fitness value as described in tables 1 to 3.

The results shown in Table 8 depict the algorithms’ convergence rate in various functions. When the number shown in Table 8 gets smaller, it shows that the algorithm has been able to perform fewer fitness evaluations in order to reach the satisfactory results. For better comparison, the algorithms are ranked in Table 9 according to the results of the Table 8. In the last row of Table 9, the sum of all algorithms ranks in all functions are calculated. As can be observed, the proposed algorithm has yielded better results in total.

The success rate (SR%) of the algorithms in 100 independent runs for each benchmark function are compared in Table 10. The success rate of the algorithms illustrates the reliability of them. In the last row of Table 10, the algorithms average success rates for all functions are shown. As can be seen, the proposed algorithm has the highest success rate among all tested algorithms. Figures 5, 6, 7 and

8 illustrate the convergence behavior of the proposed algorithm and nine other algorithms on functions f_6, f_{10}, f_{11}, f_{12}, f_{19}, f_{20}, f_{27}, and f_{28}.

Table 8: Average and standard deviation of algorithms' convergence speed.

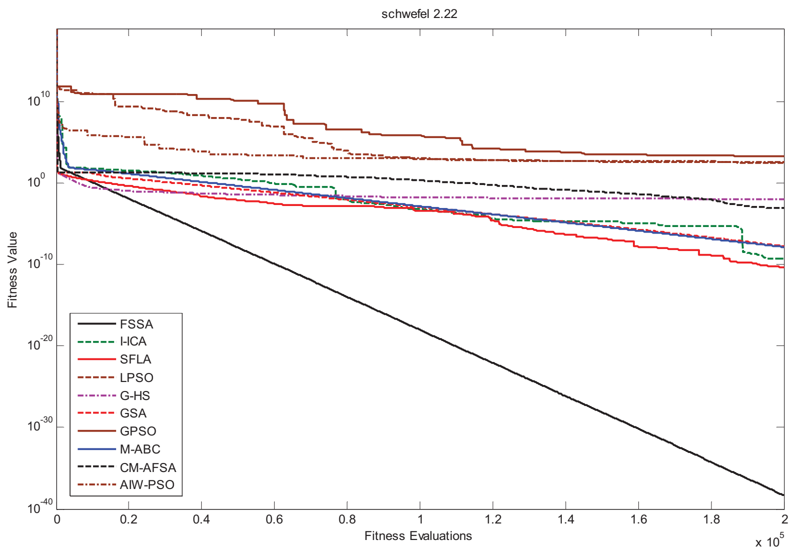
F	Algorithms									
	G-PSO	L-PSO	G-HS	I-ICA	GSA	M-ABC	CM-AFSA	SFLA	AIW-PSO	FSSA
f_1	162269.44 (745.03)	186115.88 (102.35)	-	181007.54 (5334.21)	-	-	-	-	**45525.6** (722.93)	89309.18 (1441.79)
f_2	13007.46 (5051.12)	19159.00 (4937.30)	45896.56 (11384.81)	1493.32 (413.42)	11595.05 (8167.09)	4888.56 (1351.68)	15009.42 (12636.81)	1400.86 (252.41)	**419.6** (117.06)	1284.16 (195.59)
f_3	181772.58 (25903.94)	-	73165.94 (28911.38)	-	38892.82 (16051.14)	159395.33 (70901.41)	126270.86 (1040.94)	**26411.84** (11233.49)	-	75538.05 (26095.64)
f_4	76364.05 (6090.66)	90465.04 (4113.42)	31826.02 (20905.35)	28505.86 (5240.07)	55255.70 (20644.58)	59293.46 (18085.14)	86303.62 (9636.71)	34657.18 (7544.81)	**5769.1** (936.12)	10592.86 (833.05)
f_5	107480.87 (15842.07)	128839.24 (14861.01)	-	86195.12 (52019.33)	36416.98 (20341.00)	179110.41 (68384.21)	154775.06 (14753.68)	-	**13952.66** (5175.54)	32666.65 (5147.91)
f_6	-	136583.00 (16591.81)	75479.78 (40549.16)	75805.16 (12192.67)	79292.90 (27094.37)	83775.74 (18352.11)	173982.52 (4072.39)	49518.60 (15164.99)	-	**19884.88 (356.80)**
f_7	115423.60 (5156.14)	134902.36 (4783.64)	**9724.04 (4606.59)**	57052.00 (22183.06)	17599.12 (8750.92)	73710.16 (30696.75)	126032.88 (64285.51)	18580.48 (10057.81)	58198.87 (23358.21)	15406.15 (7831.03)
f_8	105580.33 (2607.10)	130470.08 (3995.66)	18294.76 (13306.90)	45261.68 (6223.62)	55319.52 (7094.53)	74106.16 (26310.96)	125895.40 (6113.66)	34220.56 (8647.03)	74354.98 (837.42)	**15896.45 (518.82)**
f_9	48965.82 (11969.53)	64921.44 (5466.28)	312.50 (88.73)	3688.26 (1133.33)	31467.26 (28147.19)	17170.66 (16076.10)	2863.38 (3130.40)	**149.50** (33.233)	2934.44 (884.43)	2188.81 (283.78)
f_{10}	172526.50 (82467.98)	-	127961.14 (53816.41)	161926.96 (8753.10)	**59905.02** (9474.56)	155009.33 (49315.63)	169700.13 (10161.94)	105639.38 (14582.82)	-	87021.96 (2335.06)
f_{11}	111467.52 (2195.82)	135874.52 (2321.15)	42299.46 (34732.25)	53407.22 (5093.02)	43079.46 (5913.22)	97030.64 (26751.48)	153347.20 (19175.26)	36084.24 (3428.76)	**8877.24** (1109.64)	20426.90 (440.72)
f_{12}	-	-	-	-	**57782.76** (20079.82)	-	-	-	-	96631.38 (14487.19)
f_{13}	49324.22 (3472.38)	58708.68 (1729.47)	-	9135.98 (444.57)	169090.14 (58615.37)	11296.90 (970.38)	196893.48 (343.74)	4941.02 (2910.42)	**1774.24** (121.36)	2506.23 (96.17)
f_{14}	48905.66 (2726.19)	56867.42 (2559.71)	-	9044.38 (576.23)	163321.50 (24521.96)	11111.88 (30446.63)	196498.38 (929.72)	4668.58 (2691.82)	**1741.28** (103.81)	2375.28 (84.47)
f_{15}	54020.70 (3875.61)	62105.54 (2837.29)	-	10458.92 (637.26)	173125.16 (42154.52)	36700.40 (19650.31)	196682.74 (353.93)	22415.16 (16062.89)	**2340.64** (251.54)	4556.10 (159.75)
f_{16}	5786.30 (3886.44)	8403.24 (5232.75)	23121.47 (28426.16)	1647.30 (604.39)	2845.05 (609.03)	3303.70 (1109.13)	10323.60 (4616.80)	1056.60 (490.15)	**409.6** (177.51)	1004.08 (206.16)
f_{17}	172.14 (135.10)	169.34 (118.62)	9384.54 (6645.02)	352.08 (301.84)	310.74 (192.84)	722.00 (186.99)	248.60 (166.14)	211.54 (152.85)	**116.96** (95.38)	310.78 (119.80)
f_{18}	5042.42 (5268.34)	7882.92 (3951.44)	5372.32 (2345.13)	1361.36 (519.23)	9224.52 (5683.19)	1665.32 (812.55)	1410.26 (673.02)	366.36 (459.23)	**268.92** (60.44)	500.31 (80.12)
f_{19}	118905.72 (4295.86)	143468.18 (3662.89)	74014.00 (37928.36)	-	62618.26 (30818.07)	116454.38 (40937.71)	162753.33 (1706.08)	111250.84 (23673.08)	**12591.00** (2307.73)	23428.36 (402.48)
f_{20}	-	-	-	-	150270.08 (49374.39)	149298.38 (18384.68)	-	-	-	**33791.86 (1530.08)**
f_{21}	114814.58 (3908.89)	146682.59 (4954.58)	46547.93 (20121.72)	60084.00 (28989.10)	23642.00 (5076.46)	154829.13 (48197.83)	188565.00 (1305.95)	81650.42 (8800.85)	**9443.92** (1986.64)	22031.28 (1350.25)
f_{22}	120837.29 (4760.15)	145847.86 (6221.07)	**10067.22 (2399.50)**	83402.27 (36423.45)	20827.91 (7614.15)	178831.48 (19373.48)	167541.66 (6019.54)	62534.56 (32698.60)	19128.96 (4357.12)	21104.26 (2338.30)
f_{23}	122743.84 (2534.71)	146109.28 (4295.11)	**15581.12 (7135.34)**	87165.08 (40548.75)	33219.86 (10464.41)	170251.14 (39916.43)	143435.79 (15477.01)	33738.62 (6310.91)	22208.24 (6398.98)	23428.71 (1650.55)
f_{24}	-	-	-	-	62403.54 (28492.08)	121390.32 (27421.91)	171387.50 (38102.33)	-	-	**22340.85 (1816.16)**
f_{25}	-	-	-	56117.00 (43059.53)	-	149948.20 (37413.88)	165807.60 (4019.37)	-	-	**20396.81 (1485.13)**
f_{26}	-	-	-	54073.72 (8118.91)	42883.92 (11053.75)	91025.64 (28674.52)	146812.40 (50853.08)	-	-	**19528.71 (1310.07)**
f_{27}	182506.33 (42358.80)	189672.50 (39091.61)	-	-	**23842.95** (4074.37)	197751.00 (49164.29)	171654.50 (29314.52)	-	-	24680.69 (3898.31)
f_{28}	126479.06 (7717.46)	151548.18 (9007.38)	-	-	33676.96 (13805.01)	127858.24 (33641.19)	155972.12 (5085.32)	125876.92 (34642.25)	65001.85 (33899.12)	**23373.81 (5604.46)**
f_{29}	-	-	-	-	62659.00 (10379.28)	149404.48 (28919.46)	174340.91 (8019.70)	-	-	**24366.10 (2160.55)**
f_{30}	-	-	-	109910.50 (46169.80)	-	-	179368.50 (9875.28)	-	-	**46445.71 (12460.90)**

Table 9: Algorithms ranking based on convergence rate

F	Algorithms									
	G-PSO	L-PSO	G-HS	I-ICA	GSA	M-ABC	CM-AFSA	SFLA	AIW-PSO	FSSA
f_1	3	5	6	4	6	6	6	6	1	2
f_2	7	9	10	4	6	5	8	3	1	2
f_3	7	8	3	8	2	6	5	1	8	4
f_4	8	10	4	3	6	7	9	5	1	2
f_5	5	6	9	4	3	8	7	9	1	2
f_6	9	7	3	4	5	6	8	2	9	1
f_7	8	10	1	5	3	7	9	4	6	2
f_8	8	10	2	4	5	6	9	3	7	1
f_9	9	10	2	6	8	7	4	1	5	3
f_{10}	8	9	4	6	1	5	7	3	9	2
f_{11}	8	9	4	6	5	7	10	3	1	2
f_{12}	3	3	3	3	1	3	3	3	3	2
f_{13}	6	7	10	4	8	5	9	3	1	2
f_{14}	6	7	10	4	8	5	9	3	1	2
f_{15}	6	7	10	3	8	5	9	4	1	2
f_{16}	7	8	10	4	5	6	9	3	**1**	2
f_{17}	3	2	10	8	6	9	5	4	1	7
f_{18}	7	9	8	4	10	6	5	2	**1**	3
f_{19}	7	8	4	10	3	6	9	5	1	2
f_{20}	4	4	4	4	3	2	4	4	4	1
f_{21}	7	8	4	5	3	9	10	6	1	2
f_{22}	7	8	1	6	3	10	9	5	2	4
f_{23}	7	9	1	6	4	10	8	5	2	3
f_{24}	5	5	5	5	2	3	4	5	5	1
f_{25}	5	5	5	2	5	3	4	5	5	1
f_{26}	6	6	6	3	2	4	5	6	6	1
f_{27}	4	5	7	7	1	6	3	7	7	2
f_{28}	5	7	9	9	2	6	8	4	3	1
f_{29}	5	5	5	5	2	3	4	5	5	1
f_{30}	4	4	4	2	4	4	3	4	4	1
Total	*184*	*210*	*164*	*148*	*130*	*175*	*202*	*123*	*103*	***63***

Table 10: Algorithm reliability (successful rate) comparisons.

F	Algorithms									
	G-PSO	L-PSO	G-HS	I-ICA	GSA	M-ABC	CM-AFSA	SFLA	AIW-PSO	FSSA
f_1	**100**	**100**	0	**100**	0	0	0	0	**100**	**100**
f_2	**100**	**100**	78	**100**	68	**100**	**100**	**100**	**100**	**100**
f_3	35	0	**100**	0	34	13	46	**100**	0	**100**
f_4	68	89	**100**	**100**	**100**	**100**	**100**	**100**	61	**100**
f_5	68	**100**	0	**100**	**100**	63	**100**	0	**100**	**100**
f_6	0	3	67	**100**	**100**	**100**	**100**	**100**	0	**100**
f_7	**100**	**100**	**100**	3	**100**	**100**	50	**100**	96	**100**
f_8	13	93	**100**	**100**	**100**	**100**	**100**	**100**	21	**100**
f_9	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_{10}	5	0	14	**100**	**100**	7	44	**100**	0	**100**
f_{11}	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_{12}	0	0	0	0	**100**	0	0	0	0	**100**
f_{13}	**100**	**100**	0	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_{14}	**100**	**100**	0	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_{15}	**100**	**100**	0	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_{16}	**100**	**100**	79	**100**	78	**100**	**100**	**100**	**100**	**100**
f_{17}	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_{18}	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_{19}	96	**100**	86	0	**100**	**100**	18	**100**	3	**100**
f_{20}	0	0	0	0	**100**	**100**	0	0	0	**100**
f_{21}	25	55	91	16	32	92	2	14	26	63
f_{22}	62	**100**	**100**	23	98	97	7	**100**	54	**100**
f_{23}	**100**	**100**	**100**	71	**100**	**100**	79	**100**	97	**100**
f_{24}	0	0	0	0	**100**	**100**	5	0	0	**100**
f_{25}	0	0	0	6	0	**100**	11	0	0	46
f_{26}	0	0	0	**100**	**100**	**100**	**100**	0	0	**100**
f_{27}	7	5	0	0	98	3	4	0	0	**100**
f_{28}	88	**100**	0	0	**100**	**100**	83	26	56	**100**
f_{29}	0	0	0	0	**100**	**100**	11	0	0	**100**
f_{30}	0	0	0	9	0	0	5	0	0	**38**
Total SR% (Std)	55.56 (45.30)	61.50 (47.76)	47.16 (47.69)	57.6 (47.86)	83.60 (33.44)	79.16 (38.97)	58.83 (44.15)	61.33 (48.43)	50.46 (46.65)	**94.90 (15.92)**



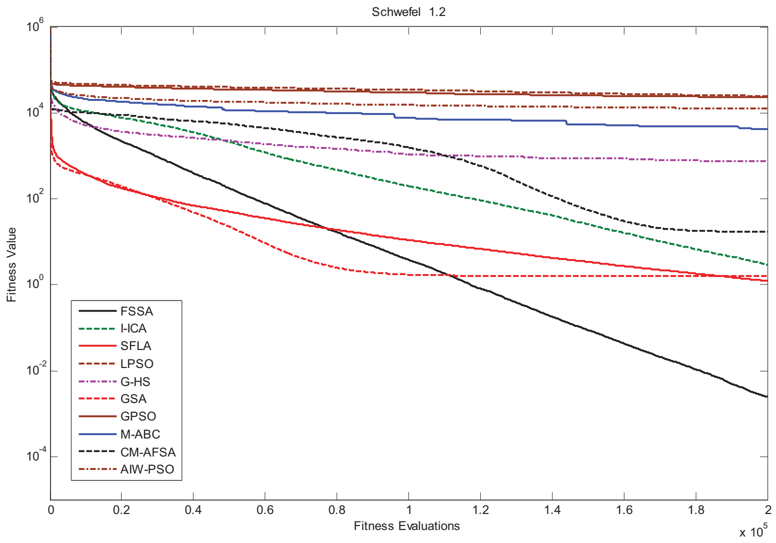
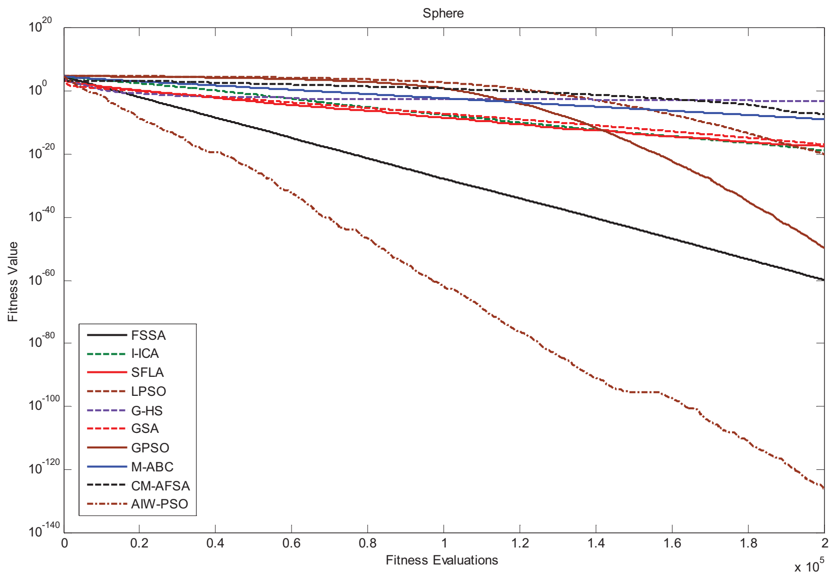


Figure 5: The graph of the algorithms convergence behavior on functions Schwefel 2.22(f_6) and Schwefel 1.2(f_{10}).



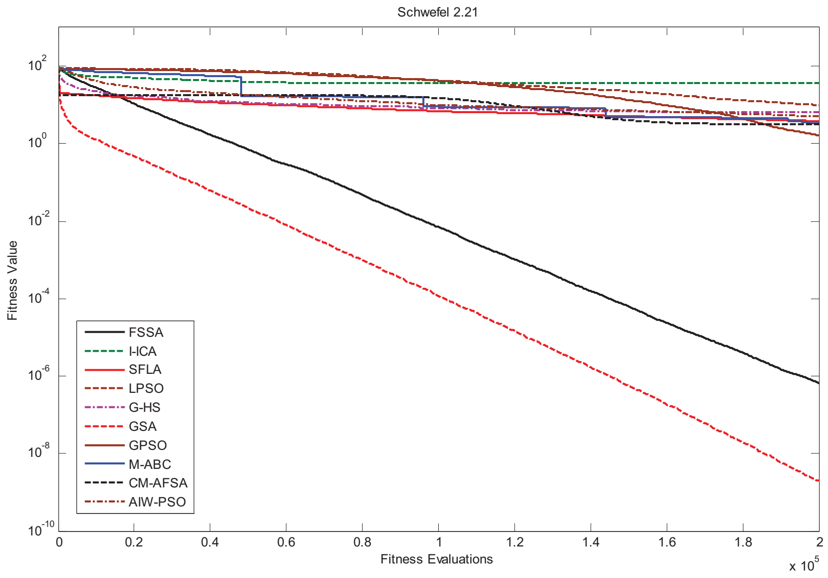
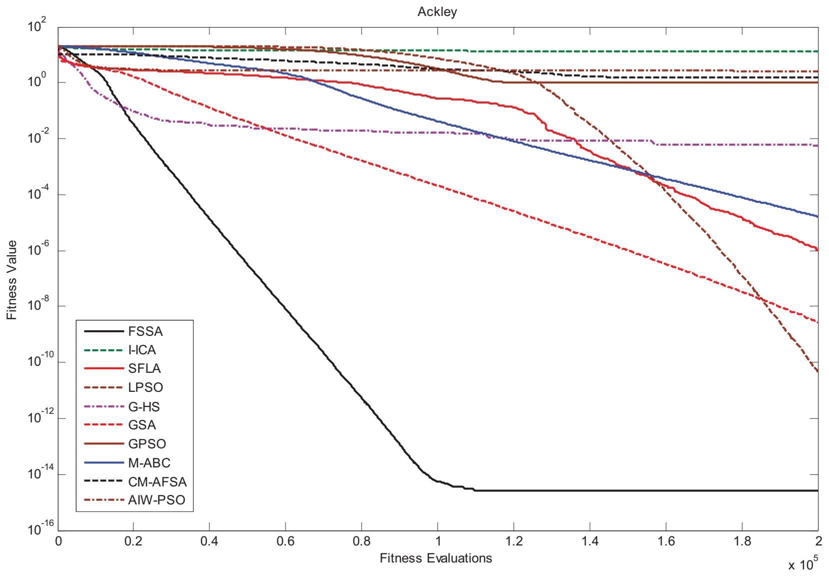


Figure 6: The graph of the algorithms convergence behavior on functions Sphere (f_{11}) and Schwefel 2.21(f_{12}).



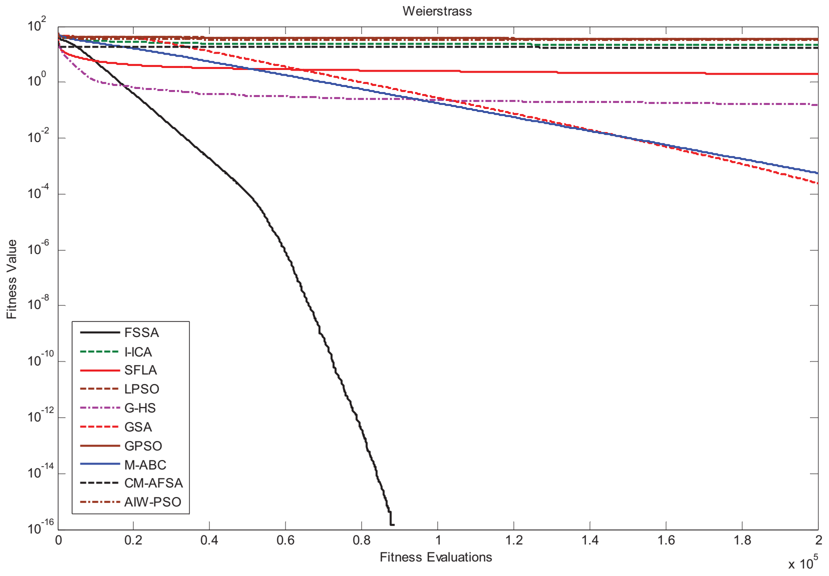
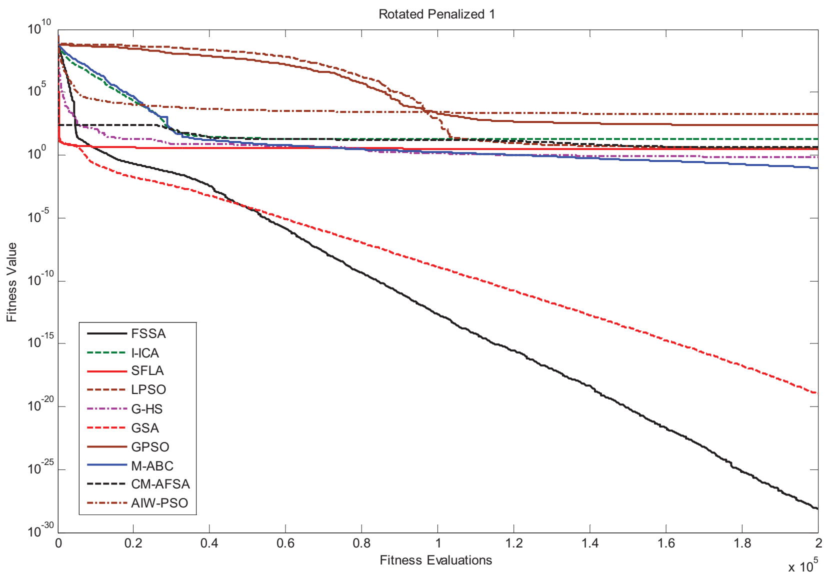


Figure 7: The graph of the algorithms convergence behavior on functions Ackley (f_{19}) and Weierstrass (f_{20}).



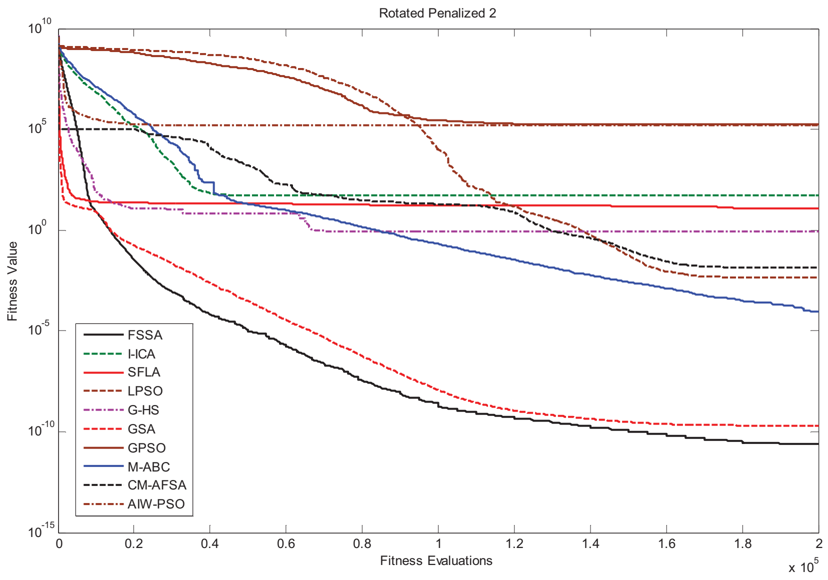


Figure 8: The graph of the algorithms convergence behavior on functions Rotated Penalized 1(f_{27}) and Rotated Penalized 2(f_{28}).

4.3. Statistical Analysis

As can be seen in Tables 6 and 7, the proposed algorithm yields to the first rank in 25 cases in which 10 cases are jointly acquired with at least one other algorithm so that it is the only algorithm which leads to the best results in 15 cases. In order to survey the significant level of the proposed algorithm in these 15 cases, compared to other comparative algorithms, a statistical test method called *one sample t-test* has been employed. H_0 and H_1 hypothesis have been defined as follows for performing the test:

$$H_0\text{: } \mu_j = a_j,^{*} \quad (7)$$

$$H_1\text{: } \mu_j \neq a_j\ ^{*} \quad (8)$$

$$a_i = \arg\min(\mu_{i,j})\ ^{*} \quad (9)$$

$$^{*}i \in \text{all algorithms, } j \in \text{all scenarios}$$

in which a_j is the average result of the best algorithm amongst comparative algorithms performed on j benchmark function and m_j is that of the proposed method performed on j benchmark function. Since, the number of samples (the number of performing algorithm) is more than 30 (n=100>30), test of normal distribution of the samples is not needed, based on the central limit theory. So that t-test could be utilized for evaluating the results. T-test approach in form of *two-tailed* test is performed with 99 degrees of freedom, where alpha is 0.01. The obtained results by t-test approach where the proposed algorithm obtained the best results (15 cases) are shown in Table 11.

Table 11: The result of T-test approach obtained from the proposed algorithm and the best comparative method.

F	Best result among other algorithms ALG.(MEAN)	FSSA Mean(Std)	Confidence interval	T_Value	h	P_Value
f_4	I-ICA(3.22e-61)	1.29e-96(2.29e-96)	[6.89e-97,1.89e-96]	-1.40e+36	1	0
f_6	SFLA(5.20e-11)	2.96e-39(3.88e-39)	[1.94e-39,3.98e-39]	-1.33e+29	1	0
f_8	I-ICA(1.11e-20)	1.41e-62(1.55e-62)	[1.00e-62,1.82e-62]	-7.15e+42	1	0
f_9	G-PSO(1.24e-85)	6.8e-126(6.5e-125)	[-1.0e-125,2.4e-125]	-1.88e+40	1	0
f_{10}	SFLA(1.23)	0.0022(0.0028)	[0.0014,0.0029]	-3608.2	1	3.2e-255
f_{19}	L-PSO(4.72e-11)	2.66e-015(0)	[2.66e-15,2.66e-15]	-Inf	1	0
f_{20}	GSA(0.0002)	0(0)	[0,0]	-Inf	1	0
f_{22}	L-PSO(1.63e-15)	1.57e-32(3.57e-47)	[1.57e-32,1.57e-32]	-4.55e+32	1	0
f_{23}	G-PSO(2.56e-30)	1.67e-33(4.17e-34)	[1.56e-33,1.78e-33]	-61302	1	0
f_{24}	GSA,M-ABC(-139.99)	-140(0)	[-140,-140]	-Inf	1	0
f_{25}	M-ABC(-179.99)	-179.99(0.005)	[-179.99,-179.99]	15.065	1	2.30e-27
f_{27}	GSA(1.10e-19)	4.31e-29(6.19e-29)	[2.68e-31,5.94e-31]	-1.77e+12	1	0
f_{28}	GSA(2.05e-010)	2.37e-11(3.63e-11)	[2.57e-11,4.7e-10]	-20.954	1	3.57e-38
f_{29}	GSA,M-ABC(-139.99)	-140(0)	[-140,-140]	-Inf	1	0
f_{30}	I-ICA,CM-AFSA(-179.91)	-179.99(0.005)	[-179.99,-179.99]	-147.45	1	7.8e-118

In Table 11 the results are tabulated for the proposed method and the best algorithm amongst other comparative algorithms and mean and standard deviation values by performing 100 executions of the proposed method are tabulated. Furthermore, designation of the best comparative algorithm along with its average results are shown. In this table, *h* indicates the rejection of H_0 hypothesis (*h*=1) or fail to rejection of H_0 hypothesis (*h*=0). T-test value, probability value and confidence interval are the output of T-test approach. As can be seen in Table 11, H_0 is rejected with 99% confidence level (1% significant level) in all functions which means that the average results obtained from the proposed algorithm is better than those obtained from the best comparative algorithm in the 15 cases with 99% confidence level.

5. CONCLUSION AND FUTURE WORK

In this paper, a novel algorithm was proposed to perform global optimization in continuous and stationary environments which was called Fish Swarm Search Algorithm. The proposed algorithm was inspired from the life of fish which live in a swarm in the nature for finding better positions in the problem space. In the proposed algorithm, the population members tried to find more food. Each fish in the swarm moved toward better fish and performed a local search in its neighborhood to find better positions. In the proposed algorithm, the fish maintained high diversity in the swarm by moving in the swarm based on the central position which increases the swarm ability to find better peaks and to escape from local optima.

Several advantages have been involved in this approach i.e. comprehensibility, simplicity, high convergence rate, high reliability, and high balance between exploration and exploitation. In the proposed algorithm, there is only one parameter to be initialized at the beginning of the process, i.e. population size that makes the algorithm considerably simpler in various applications. The proposed algorithm was evaluated on 30 benchmarks including unimodal, multimodal, shifted and rotated functions and has been compared with nine other well-known algorithms in this domain. The results of the experiments and the analysis showed that the proposed outperformed other comparative studies, in term of performance and reliability.

The experiments results and the evaluations showed that no algorithm may conquer all problems. In fact, some algorithms may either perform well in some problems or have an unsatisfactory performance in others. The proposed algorithm is no exception, and may not be performed well in some problems. More precisely, in this research, only stationary continuous environments were considered. Modifying the proposed algorithm to be utilized in other optimization types such as discrete, multi-objective and dynamic environments optimization and applying the proposed algorithm for real world applications such as engineering optimization problems can be pursued in the future works. In addition, as it was seen, suitable distribution of the initial population is considered among the important criteria for success of an algorithm. However, distribution of the initial population is a random process in most swarm intelligence algorithms which may lead to different results. In this paper, we also considered random initial population at the start of the algorithm, however, we will survey the strategies for distributing the initial fish and investigate the effects of such distributions in future work.

REFERENCES

Akay, B., Karaboga, D., 2012, A modified Artificial Bee Colony algorithm for real-parameter optimization. *Information Sciences* **192**, 120-142, DOI: 10.1016/j.ins.2010.07.015.

Ali, M., Alkhatib, K., Tashtoush, Y., 2013, Cultural algorithms: Emerging social structures for the solution of complex optimization problems. *International Journal of Artificial Intelligence* **11**, 20-42.

Amiri, B., Fathian, M., Maroosi, A., 2009, Application of shuffled frog-leaping algorithm on clustering. *The International Journal of Advanced Manufacturing Technology* **45**, 199-209, DOI: 10.1007/s00170-009-1958-2.

Atashpaz-Gargari, E., Lucas, C., 2007, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. *IEEE Congress on Evolutionary computation*, Singapore*,* 4661-4667, DOI: 10.1109/CEC.2007.4425083.

Bahrololoum, A., Nezamabadi-pour, H., Bahrololoum, H., Saeed, M., 2012, A prototype classifier based on gravitational search algorithm. *Applied Soft Computing* **12**, 819-825, DOI: 10.1016/j.asoc.2011.10.008.

Chen, L.F., Su, C.T., Chen, K.H., Wang, P.C., 2012, Particle swarm optimization for feature selection with application in obstructive sleep apnea diagnosis. *Neural Computing and Applications* **21**, 2087-2096, DOI: 10.1007/s00521-011-0632-4.

Chung, G., Lansey, K., 2009, Application of the shuffled frog leaping algorithm for the optimization of a general large-scale water supply system. *Water Resources Management* **23**, 797-823, DOI: 10.1007/s11269-008-9300-6.

Coelho, L.D.S., Afonso, L.D., Alotto, P., 2012, A modified imperialist competitive algorithm for optimization in electromagnetics. *IEEE Transactions on Magnetics* **48**, 579-582, DOI: 10.1109/TMAG.2011.2172400.

Duan, H., Xu, C., Liu, S., Shao, S., 2010, Template matching using chaotic imperialist competitive algorithm. *Pattern Recognition Letters* **31**, 1868-1875, DOI: 10.1016/j.patrec.2009.12.005.

Duman, S., Guvenc, U., Sonmez, Y., Yorukeren, N., 2012, Optimal power flow using gravitational search algorithm. *Energy Conversion and Management* **59**, 86-95, DOI: 10.1016/j.enconman.2012.02.024.

Ebrahimi, J., Hosseinian, S.H., Gharehpetian, G.B., 2011, Unit commitment problem solution using shuffled frog leaping algorithm. *IEEE Transactions on Power Systems* **26**, 573-581, DOI: 10.1109/TPWRS.2010.2052639.

Eusuff, M., Lansey, K., 2003, Optimization of water distribution network design using th shuffled frog leaping algorithm. *Journal of Water Resources Planning and Management* **129**, 210-225, DOI: 10.1061/(ASCE)0733-9496(2003)129:3(210)).

Forsati, R., Mahdavi, M., Shamsfard, M., Meybodi, M.R., 2012, Efficient stochastic algorithms for document clustering. *Information Sciences*, DOI: 10.1016/j.ins.2012.07.025.

Huang, T., Mohan, A.S., 2005, A hybrid boundary condition for robust particle swarm optimization. *IEEE Antenna and Wireless Propagation Letters* **4**, 112-117.

Ji, J., Wei, H., Liu, C., 2013, An artificial bee colony algorithm for learning Bayesian networks. *Soft Computing* **17**, 983-994, DOI: 10.1007/s00500-012-0966-6.

Jiao, B., Yan, S., 2011, A Cooperative Co-evolutionary Quantum Particle Swarm Optimizer based on Simulated Annealing for Job Shop Scheduling Problem. *International Journal of Artificial Intelligence* **7**, 232-247.

Kang, F., Li., J., Xu, Q., 2012, Damage detection based on improved particle swarm optimization using vibration data. *Apllied Soft Computing* **12**, 2329-2335, DOI: 10.1016/j.asoc.2012.03.050.

Karaboga, D., 2005, An idea based on honey bee swarm for numerical optimization. *Technical Report TR06*, Computer Engineering Department, Engineering Faculty, Erciyes University, Turkey.

Karaboga, D., Akay, B., 2009, A survey: Algorithms simulating bee swarm intelligence. *Artificial Intelligence Review* **31**, 61-85, DOI: 10.1007/s10462-009-9127-4.

Karaboga, D., Akay, B., 2011, A modified artificial bee colony (ABC) algorithm for constrained optimization problems. *Applied Soft Computing* **11**, 3021-3031, DOI: 10.1016/j.asoc.2010.12.001.

Karaboga, D., Ozturk, C., 2011, A novel clustering approach: artificial bee colony (ABC) algorithm. *Applied Soft Computing* **11**, 652-657, DOI: 10.1016/j.asoc.2009.12.025.

Karaboga, D., Ozturk, C., Karaboga, N., Gorkemli, B., 2012, Artificial bee colony programming for symbolic regression. *Information Sciences* **209**, 1-15, DOI: 10.1016/j.ins.2012.05.002.

Kasabov, N., Hamed, H.N.A., 2011, Quantum-inspired Particle Swarm Optimisation for Integrated Feature and Parameter Optimisation of Evolving Spiking Neural Networks. *International Journal of Artificial Intelligence* **7,** 114-124.

Kaveh, A., Talatahari, S., 2010, Optimum design of skeletal structures using imperialist competitive algorithm. *Computers & Structures* **88**, 1220-1229, DOI: 10.1016/j.compstruc.2010.06.011.

Kennedy, J., Eberhart, R.C., 1995, Particle swarm optimization. *IEEE International Conference on Neural Networks,* Perth, WA, **4**, 1942-1948, DOI: 10.1109/ICNN.1995.488968.

Kennedy, J., Mondes, R., 2002, Population structure and particle swarm performance. *IEEE Congress on Evolutionary Computation,* Honolulu, HI, **2**, 1671-1676, DOI:10.1109/CEC.2002.1004493.

Khajehzadeh, M., Taha, M.R., El-Shafie, A., Eslami, M., 2012, A modified gravitational search algorithm for slope stability analysis. *Engineering Applications of Artificial Intelligence* **25**, 1589–1597, 10.1016/j.engappai.2012.01.011.

Lei, L.X., Shao, Z.J., Qian, J.X., 2002, An optimizing method based on autonomous animate: Fish swarm algorithm. *System Engineering Theory and Practice* **11**, 32-38.

Li, C., Yang, S., Nguyen, T.T., 2012, A self-learning particle swarm optimizer for global optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* **42**, 627-646, DOI: 10.1109/TSMCB.2011.2171946.

Li, J., Pan, Q., Xie, S., 2012, An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems. *Applied Mathematics and Computation* **218**, 9353-9371, DOI: 10.1016/j.amc.2012.03.018.

Li, X., Yao, X., 2012, Cooperatively coevolving particle swarms fo large scale optimization. *IEEE Transactions on Evolutionary Computation* **16**, 210-224, DOI: 10.1109/TEVC.2011.2112662.

Li, X., Yin, M., 2012, Self-adaptive constrained artificial bee colony for constrained numerical optimization. *Neural Computing and Applications* **218***,* 10943–10973*,* DOI: 10.1007/s00521-012-1285-7.

Li, Y., Liang, J., 2011, A Hybrid Multi-swarm Co-evolutional Particle Swarm Optimizer. *International Journal of Artificial Intelligence* **7,** 274-287.

Liu, J., Ren, X., Ma, H., 2012, Adaptive swarm optimization for locating and tracking multiple targets. *Applied Soft Computing* **12**, 3656-3670, DOI: 10.1016/j.asoc.2012.06.005.

Ma, M., Liang, J., Guo, M., Fan, Y., Yin, Y., 2011, SAR image segmentation based on Artificial Bee Colony algorithm*. Applied Soft Computing* **11**, 5205-5214, DOI: 10.1016/j.asoc.2011.05.039.

Miguel, L., Miguel, L., Jr, J., Riera, J., 2012, Damage detection under ambient vibration by harmony search algorithm. *Expert Systems with Applications* **39**, 9704-9714, DOI: 10.1016/j.eswa.2012.02.147.

Mohammadi-ivatloo, B., Rabiee, A., Soroudi, A., Ehsan, M., 2012, Imperialist competitive algorithm for solving non-convex dynamic economic power dispatch. *Energy* **44**, 228-240, DOI: 10.1016/j.energy.2012.06.034.

Nickabadi, A., Ebadzadeh, M.M., Safabakhsh, R., 2011, A novel particle swarm optimization algorithm with adaptive inertia weight. *Applied Soft Computing* **11**, 3658-3670, DOI: 10.1016/j.asoc.2011.01.037.

Niknam, T., Firouzi, B.B., Mojjarad, H.D., 2011, A new evolutionary algorithm for non-linear economic dispatch. *Expert Systems with Applications* **38**, 13301-13309, DOI: 10.1016/j.eswa.2011.04.151.

Nolle, L., Zelinka, I., Hopgood, A.A., Goodyear, A., 2005, Comparison of a self-organizing migration algorithm with simulated annealing and differential evolution for automated waveform tuning. *Advances in Engineering Software* **36**, 645-653, DOI:10.1016/j.advengsoft.2005.03.012.

Omran, M., Mahdavi, M., 2007, Global-best harmony search. *Applied Mathematics and Computation* **198**, 643-656, DOI: 10.1016/j.amc.2007.09.004.

Pan, Q., Suganthan, P.N., Tasgetiren, M., Liang, J.J., 2010, A self-adaptive global best harmony search algorithm for continuous optimization problems. *Applied Mathematics and Computation* **216**, 830-848, DOI: 10.1016/j.amc.2010.01.088.

Pan, Q., Wang, L., Gao, L., 2011, A chaotic harmony search algorithm for the flow shop scheduling problem with limited buffers. *Applied Soft Computing* **11**, 5270-5280, DOI: 10.1016/j.asoc.2011.05.033.

Pandi, V., Panihrahi, B., 2011, Dynamic economic load dispatch using hybrid swarm intelligence based harmony search algorithm. *Expert Systems with Applications* **38**, 8509-8514, DOI: 10.1016/j.eswa.2011.01.050.

Poli, R., Kenedy, J., Blackwell, T., 2007, Particle swarm optimization: An overview. *Swarm Intelligence* **1**, 33-57, DOI:10.1007/s11721-007-0002-0.

Purcaru, C., Precup, R.E., Iercan, D., Fedorovici, L.O., David, R.C., Dragan, F., 2013, Optimal robot path planning using gravitational search algorithm. *International Journal of Artificial Intelligence* **10,** 1-20.

Precup, R.-E., David, R.-C., Petriu, E.M., Preitl, S., Paul, A.S., 2011, Gravitational search algorithm-based tuning of fuzzy control systems with a reduced parametric sensitivity. In *Soft Computing in Industrial Applications*, A. Gaspar-Cunha, R. Takahashi, G. Schaefer, L. Costa, Eds. Springer-Verlag, Berlin, Heidelberg, Advances in Intelligent and Soft Computing **96**, 141-150, DOI: 10.1007/978-3-642-20505-7_12.

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2009, GSA: A gravitational search algorithm. *Information Sciences* **179**, 2232-2248, DOI: 10.1016/j.ins.2009.03.004.

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2010, BGSA: Binary gravitational search algorithm. *Natural Computing* **9**, 727-745, DOI: 10.1007/s11047-009-9175-3.

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2011, Filter modeling using gravitational search algorithm. *Engineering Applications of Artificial Intelligence* **24**, 117-122, DOI: 10.1016/j.engappai.2010.05.007.

Rashtchi, V., Rahimpour, E., Shahrouzi, H., 2012, Model reduction of transformer detailed R-C-L-M model using the imperialist competitive algorithm. *IET Electric Power Applications* **6**, 233-242, DOI: 10.1049/iet-epa.2011.0331.

Robinson, J., Rahmat-Samii, Y., 2004, Particle swarm optimization in electromagnetics. *IEEE Transaction on Antennas and Propagation* **52**, 397-407, DOI: 10.1109/TAP.2004.823969.

Rocha, A., Martins, T., Fernandes, E., 2011, An augmented lagrangian fish swarm based method for global optimization. *Journal of Computational and Applied Mathematics* **235**, 4611-4620, DOI: 10.1016/j.cam.2010.04.020.

Saha, S.K., Ghoshal, S.P., Kar, R., Mandal, D., 2013, Cat swarm optimization algorithm for optimal linear phase FIR filter design. *ISA Transactions* **52**, 781-794, DOI: 10.1016/j.isatra.2013.07.009.

Shen, W., Guo, X., Wu, C., Wu, D., 2011, Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. *Knowledge-Based Systems* **24**, 378-385, DOI: 10.1016/j.knosys.2010.11.001.

Shi, Y., Eberhart, R.C., 1998, A modified particle swarm optimizer. *IEEE World Congress on Computational Intelligence*, Anchorage, AK, USA, 69-73, DOI: 10.1109/ICEC.1998.699146.

Shirkouhi, S.N., Eyvazi, H., Ghodsi, K., Atashparz-G, E., 2010, Solving the integrated product mix-outsourcing problem using the imperialist competitive algorithm. *Expert Systems with Applications* **37**, 7615-7626, DOI: 10.1016/j.eswa.2010.04.081.

Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S., 2005, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *IEEE Congress on Evolutionary Computation*, 1-50.

Tsai, H.C., Lin, Y.H., 2011, Modification of the fish swarm algorithm with particle swarm optimization formulation and communication behavior. *Applied Soft Computing* **11**, 5367–5374, DOI: 10.1016/j.asoc.2011.05.022.

Wang, H., Yuan, X., Wang, Y., Yang, Y., 2013, Harmony search algorithm-based fuzzy-PID controller for electronic throttle valve. *Neural Computing and Applications* **22**, 329-336, DOI: 10.1007/s00521-011-0678-3.

Xu, L., Fei, M., Jia, T., Yang, T.C., 2012, Bandwidth scheduling and optimization using non-cooperative game model-based shuffled frog leaping algorithm in a networked learning control system. *Neural Computing and Applications* **21**, 1117-1128, DOI: 10.1007/s00521-011-0736-x.

Yang, S., Li, C., 2010, A clustering particle swarm optimization for locating and tracking optima in dynamic environments. *IEEE Transactions on Evolutionary Computation* **14**, 959-974, DOI: 10.1109/TEVC.2010.2046667.

Yang, X.S., Cui, Z., Xiao, R., Gandomi, A.H., Karamanoglu, M., 2013, Swarm intelligence and bio-inspired computation: theory and applications. *Elsevier*, ISBN: 978-0-12-405163-8.

Yao, X., Liu, Y., Lin, G., 1999, Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation* **3**, 82-102, DOI: 10.1109/4235.771163.

Yazdani, D., Akbarzadeh-T, M.R., Nasiri, B., Meybodi, M.R., 2012, A new artificial fish swarm algorithm for dynamic optimization problems. *IEEE Congress on Evolutionary Computation (CEC2012)*, Brisbane, QLD, 1-8, DOI: 10.1109/CEC.2012.6256169.

Yazdani, D., Arabshahi, A., Sepas-Moghaddam, A., Dehshibi, M.M., 2012, A multilevel thresholding method for image segmentation using a novel hybrid intelligent approach. *12th International Conference on Hybrid Intelligent Systems (HIS)*, Pune,137-142, DOI: 10.1109/HIS.2012.6421323.

Yazdani, D., Golyari, S., Meybodi, M.R., 2010, A new hybrid algorithm for optimization based on artificial fish swarm algorithm and cellular learning automata. *5th International Symposium on Telecommunications (IST)*, Tehran, 914-919, DOI: 10.1109/ISTEL.2010.5734153.

Yazdani, D., Golyari, S., Meybodi, M.R., 2010, A new hybrid approach for data clustering. *5th International Symposium on Telecommunications (IST)*, Tehran, 932-937, DOI: 10.1109/ISTEL.2010.5734156.

Yazdani, D., Nabizadeh, H., Kosari, E.M., Toosi, A.N., 2011, Color quantization using modified artificial fish swarm algorithm. *Lecture Notes in Computer Science* **7106**, 382-391, DOI: 10.1007/978-3-642-25832-9_39.

Yazdani, D. Nadjaran Toosi, A., Meybodi, M.R., 2011, Fuzzy adaptive artificial fish swarm algorithm. In *AI 2010: Advances in Artificial Intelligence*, Springer-Verlag, Lecture Notes in Computer Science **6464**, 334-343, DOI: 10.1007/978-3-642-17432-2_34.

Yazdani, D., Nasiri, B., Azizi, R., Sepas-Moghaddam, A., Meybodi, M.R., 2013, Optimization in dynamic environments utilizing a novel method based on particle swarm optimization. *International Journal of Artificial Intelligence* **11**, 170-192.

Yazdani, D., Nasiri, B., Sepas-Moghaddam, A., Meybodi, M.R., 2013, A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization. *Applied Soft Computing* **13**, 2144-2158, DOI: 10.1016/j.asoc.2012.12.020.

Yazdani, D., Saman, B., Sepas-Moghaddam, A., Kazemi, F.M., Meybodi, M.R., 2013, A new algorithm based on improved artificial fish swarm algorithm for data clustering. *International Journal of Artificial Intelligence* **11**, 193-221.

Zhan, Z., Zhang, J., Li, Y., Shi, Y., 2011, Orthogonal particle swarm optimization. *IEEE Transactions on Evolutionary Computation* **15**, 832-847, DOI: 10.1109/TEVC.2010.2052054.

Zhang, J., Chung, H.S.H., Zhong, W., Wu, W., Shi, Y., 2010, A novel set-based particle swarm optimization method for discrete optimization problems. *IEEE Transactions on Evolutionary Computation* **14**, 278-300, DOI: 10.1109/TEVC.2009.2030331.

Zhao, W., 2011, Adaptive image enhancement based on gravitational search algorithm. *CEIS 2011, Procedia Engineering* **15**, 3288-3292, DOI: 10.1016/j.proeng.2011.08.617.

Zheng, G., Lin, Z., 2012, A winner determination algorithm for combinatorial auctions based on hybrid artificial fish swarm algorithm. *Physics Procedia, International Conference on Solid State Devices and Materials Science* **25**, 1666-1670, DOI: 10.1016/j.phpro.2012.03.292.

Zhu, W., Jiang, J., Song, C., Bao, L., 2012, Clustering algorithm based on fuzzy c-means and artificial fish swarm. *Procedia Engineering, 2012 International Workshop on Information and Electronics Engineering* **29**, 3307-3311, DOI: 10.1016/j.proeng.2012.01.485.

Zou, D., Gao, L., Li, S., Wu, J., 2011, Solving 0–1 knapsack problem by a novel global harmony search algorithm. *Applied Soft Computing* **11**, 1556-1564, DOI: 10.1016/j.asoc.2010.07.019.