Expert Systems With Applications 102 (2018) 234–244

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Neural network and fuzzy system for the tuning of Gravitational Search Algorithm parameters

Danilo Pelusi[a,*], Raffaele Mascella[a], Luca Tallini[a], Janmenjoy Nayak[b], Bighnaraj Naik[c], Ajith Abraham[d]

[a] Faculty of Communications Sciences, University of Teramo, Italy
[b] Department of Computer Sc. Engg., Sri Sivani College of Engineering, Srikakulam, India
[c] Department of Computer Application Veer Surendra Sai University of Technology, Odisha, India
[d] Machine Intelligence Research Labs (MIR Labs) Scientific Network for Innovation and Research Excellence Auburn, Washington, USA

ARTICLE INFO

Article history:
Received 12 October 2017
Revised 16 February 2018
Accepted 17 February 2018
Available online 17 February 2018

Keywords:
Computational intelligence
Fuzzy systems
Gravitational Search Algorithm
Neural networks
Search algorithms.

ABSTRACT

A good trade-off between exploration and exploitation to find optimal values in search algorithms is very hard to achieve. On the other hand, the combination of search methods may cause computational complexity increase problems. The Gravitational Search Algorithm (GSA) is a swarm optimization algorithm based on the law of gravity, where the solution search process depends on the velocity of particles. The application of intelligent techniques can improve the search performances of GSA. This paper proposes the design of a Neuro and Fuzzy Gravitational Search Algorithm (NFGSA) to achieve better results than GSA in terms of global optimum search capability and convergence speed, without increasing the computational complexity. Both the algorithms have the same computational complexity $O(nd)$, where n is the number of agents and d is the search space dimension. The main task of the designed intelligent system is to adjust a GSA parameter on a revised version of GSA. NFGSA is compared with GSA, a Plane Surface Gravitational Search Algorithm (PSGSA) and a Modified Gravitational Search Algorithm (MGSA). The results show that NFGSA improves the optimization performances of GSA and PSGSA, without adding computational costs. Moreover, the proposed algorithm is better than MGSA for a benchmark function and achieves similar results for two test functions. The analysis on the computational complexity shows that NFGSA has a better computational complexity than MGSA, because NFGSA has complexity $O(nd)$, whereas MGSA has complexity $O((nd)^2)$.

1. Introduction

Solving optimization problems using exhaustive search techniques is not a practicable way. The main problem is that the search space increases hugely according to problem size. However, difficult real-world engineering problems can be solved by using algorithms inspired by nature such as Genetic Algorithms (GA) (Holland, 1992), Particle Swarm Optimization (PSO) (Clerc, 2006), Differential Evolution (DE) (Storn & Price, 1997), Central Force Optimization (CFO) (Formato, 2007; 2008). For these stochastic algorithms, the search may start either from a single point (Kirkpatrick, Gelatto, & Vecchi, 1983) or in a parallel way with more than one initial point (Kennedy & Eberhart, 2001), (Engelbrecht, 2005).

The Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-pour, & Saryazdi, 2009) is a swarm intelligence type algorithm based on the Newton's law of universal gravitation and motion of individuals in nature (Holliday, Resnick, & Walker, 1993; Schutz, 2003). There is a tendency for gravitational force to increase with higher masses product, therefore agents with heavier masses attract more strongly than lower masses individuals. Moreover, individuals with large masses move slowly respect to individuals with low masses. GSA is an optimization algorithm that assures a good compromise between exploration and exploitation. Exploration consists of probing a portion of the search space with the hope of finding promising solutions: the aim is to avoid getting trapped in a local optimum. Exploitation consists of probing a limited, but promising, region of the search space with the hope of improving the promising solution that comes from exploration. Therefore, in GSA, lighter

* Corresponding author.
E-mail addresses: dpelusi@unite.it (D. Pelusi), rmascella@unite.it (R. Mascella), ltallini@unite.it (L. Tallini), mailforjnayak@gmail.com (J. Nayak), mailtobnaik@gmail.com (B. Naik), ajith.abraham@ieee.org (A. Abraham).

https://doi.org/10.1016/j.eswa.2018.02.026
0957-4174/

masses individuals are responsible for the exploration of the search space, whereas exploitation depends on heavier masses individuals. At the beginning of the search process, individuals are far from the optimal solution and those ones with lighter masses move with large step size (exploration). At the end of the search process, individuals converge to the optimum solution thanks to higher masses individuals which move with small step size (exploitation).

The task of improving exploration and exploitation was achieved by Saeidi-Khabisi and Rashedi (2012) with a Fuzzy Gravitational Search Algorithm (FGSA). The main idea of their approach was the fuzzy tuning of GSA parameters. Sombra, Valdez, Melin, and Castillo (2013) used type-1 fuzzy techniques to adjust GSA parameters to provide a suitable acceleration to each agent to improve its performance. Further improvements were achieved by Gonzalez, Valdez, and Melin (2016) using type-2 fuzzy logic. FGSA has been also designed as classification method for decision function estimation (Askari & Zahiri, 2011). A fuzzy-dynamic parameter adaptation of GSA has been proposed by Olivas, Valdez, and Ostillo (2016). Rashedi, Nezamabadi-pour, and Saryazdi (2010) proposed a binary version of GSA, named BGSA, particularly efficient for solving some nonlinear test functions. To solve combinatorial optimization problems, Dowlatshahi, Nezamabadi-pour, and Mashinchi (2014) proposed a Discrete Gravitational Search Algorithm (DGSA). An improvement of GSA has been proposed by Altinoz, Yilmaz, and Weber (2014): the novelty consists of generating the initial position of the population in a suitable quasi-random manner. Relevant results have been achieved by Gupta, Sharma, and Sharma (2017) with a variant of GSA, called Exploitative Gravitational Search Algorithm (EGSA), able to find a good compromise between exploration and exploitation of the search space. To improve convergence and exploration and exploitation capability of GSA, a Fitness Based Gravitational Search Algorithm (FBGSA) has been proposed (Gupta, Sharma, & Sharma, 2016a). Khajehzadeh, Taha, EI-Shafie, and Eslami (2012) introduced a constraint on velocity, which aims to check the global exploration ability of GSA. Pelusi et al. proposed optimal fuzzy systems to improve the search performances of GSA (Pelusi, Mascella, & Tallini, 2017). A Modified Gravitational Search Algorithm (MGSA) was proposed by Li, Chang, Huang, Liu, and Zhang (2016) with the definition of a novel constant attenuation factor and the use of DE mutation and crossover operators to generate new agents for increasing the population diversity. In Gupta, Sharma, and Sharma (2016b), the velocity update process of GSA is modified such that during the first iterations agents move with large step size to explore the search space. However, the idea to control the search procedure of nature-based algorithms via fuzzy controllers, has been successfully accomplished for some evolutionary and swarm intelligent techniques (Eiben, Hinterding, & Michalewicz, 1999; Setnes & Roubos, 2000; Shi, Eberhart, & Chen, 1999). Some intelligent evolutionary optimization methods use experts knowledge to deduce optimal parameters to obtain the evolution in an intelligent manner (Montiel, Castillo, Melin, Rodriguez Diaz, & Sepulveda, 2007). Relevant results on the issue of optimizing Fuzzy Inference Systems by using evolutionary approaches have been achieved by Castillo, Valdez, and Melin (2007). On the other hand, Peraza, Valdez, Garcia, Melin, and Castillo (2016) proposed a dynamic adaptation of the harmony search algorithm parameters. Finally, a population-based search algorithm has the capability of solving some problems rather than others (Wolpert & Macready, 1997): this means that no heuristic algorithm always assures a better search performance than others.

The capability of adjusting the parameters of optimization algorithms can be increased with the help of Neural Networks (NN) (Hassoun & H., 1995). NN are data driven self-adaptive methods able to learn from data (Haykin, 1998) and approximate any function with arbitrary accuracy (Hornik, 1991). These adaptive techniques can be combined with fuzzy logic to design hybrid intelligent systems able to solve different problems. As an example, Ozan Kocadagli and Langari (2017) designed an Artificial Neural Network (ANN), powered by fuzzy relations, for the classification of Electroencephalogram (EEG) signals for early detection of epileptic seizures. Bidirectional Associative Memory (BAM) neural networks and fuzzy inference system have been proposed to design a robust video watermarking system (Loganathan & Kaliyaperumal, 2016). The combination between neural learning and fuzzy learning concepts has been addressed in Zhou, Chen, and Wang (2014), where the proposed algorithm first transforms the original data into a latent space using deep learning and then fuzzifies the deep representation at the output layer for patten classification. Doulamis et al. (2004) proposed a combined fuzzy classification and neural network model for Non-Linear Prediction of 3-D Rendering Workload in Grid Computing. In this approach, the fuzzy classification is used for organizing rendering descriptors, while neural network for modeling and predicting the rendering workload. A combination of Recurrent Neural Networks (RNN) and fuzzy rules has been accomplished for graph matching (Krleza & Fertalj, 2017). Mendoza, Melin, and Licea (2009) used a type-2 fuzzy system for a pre-processing applied to the training data for better use in the NN. NN have been also combined with GSA to obtain intelligent hybrid systems for various application fields (Fedorovici, Precup, Dragan, David, & Purcaru, 2012; Ghalambaz et al., 2011). Fedorovici et al. proposed an embedding GSA in Convolutional Neural Networks (CNN) for Optical Character Recognition (OCR) applications. GSA has been also applied to train a multilayer perceptron neural network used as approximation solution of the Wessinger's equation (Ghalambaz et al., 2011). In order to optimize modular neural networks in echocardiogram recognition, a GSA with fuzzy-dynamic parameter adaptation has been proposed (Gonzalez, Valdez, Melin, & Prado-Arechiga, 2015). Because our approach consists of a dynamic regulation of parameters, the use of NN is suitable for their data-based auto-adjusting features.

In this paper, a revised GSA with intelligent adaptation of parameters is proposed. Some parameters have a big impact in the performance and quality of GSA results. The idea is to find the best way to improve the GSA abilities to perform local and global search. In other words, it needs to have optimal exploration and exploitation during the search phase. This can be made by adjusting the most relevant parameters of GSA through neural networks and fuzzy systems. NN has the task to supply the data-based ranges of GSA parameters to the fuzzy system, whereas the fuzzy system avoids the trapping in local optima and premature convergence. The main contribution of the work includes the definition of a new quantity to describe the population progress, a redefinition of a GSA key parameter and the combination of a neural network with a fuzzy system. The designed intelligent system does not add computational complexity to GSA and takes into account population progress, iterations number and some GSA features. In this way, a Neuro and Fuzzy Gravitational Search Algorithm (NFGSA) is designed. The final goal is to improve the performance of GSA in terms of optimal search capability and convergence speed, without adding any computational cost. A comparison with MGSA (Li et al., 2016) in terms of computational complexity analysis is also proposed. Moreover, NFGSA is compared with an algorithm which calculates dynamically GSA parameters by means of mathematical function.

The Section 2 contains the preliminaries of GSA. The design of NFGSA is described in the Section 3. The experimental results are discussed in the Section 4, whereas the Section 5 concludes the paper and uncovers the possible future extensions of the paper.

2. Preliminaries

In order to describe our algorithm, some GSA preliminaries are needed. To support the understanding of our approach, the same mathematician formalism of Rashedi et al. (2009) is used.

The fundamental interactions (Schutz, 2003) of the nature are the electromagnetic, weak, strong and gravitational interactions (Holliday et al., 1993; Schutz, 2003). The Eq. (1) defines the gravitational interaction between the masses M_1 and M_2 placed at distance R, with G gravitational constant. Note that, the higher the product of the masses, the higher the tendency for F to increase and a slower distance between the particles tends to cause an increase of F.

$$F = G\frac{M_1 M_2}{R^2} \tag{1}$$

The second law of Newton (Holliday et al., 1993) states that the acceleration a is equal to the ratio between F and M (see (2)).

$$a = \frac{F}{M} \tag{2}$$

Now, it needs the definition of various masses kinds: active, passive and inertial (Rashedi et al., 2009).

Let $X_i = (x_i^1, \ldots, x_i^d, \ldots, x_i^n)$ be the position of the ith agent for $i = 1, 2, \ldots, n$; x_i^d is the position of ith agent in the dth dimension. The quantity $F_{ij}^d(t)$ (see (3)) represents the force applied on mass i from mass j at time t. Moreover, $G(t)$ is the gravitational constant, M_{aj} and M_{pi} are the active and passive gravitational masses related to agent j and i respectively, ϵ is a small constant and $R_{ij}(t)$, as defined in (4), is the Euclidian distance between agents i and j.

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) \times M_{ai}(t)}{R_{ij}(t) + \epsilon}(x_j^d(t) - x_i^d(t)) \tag{3}$$

$$R_{ij}(t) = ||X_i(t), X_j(t)||_2 \tag{4}$$

The effect of decreasing gravity causes the decrease of the gravitational constant G which depends on the actual age of the universe. The decrease of the gravitational constant is defined by Eq. (5)

$$G(t) = G(t_0) \times \left(\frac{t_0}{t}\right)^{\beta} \tag{5}$$

where $G(t)$ is the value of the gravitational constant at time t. $G(t_0)$ is the value of the gravitational constant at the first cosmic quantum-interval of time t_0 (Mansouri, Nasseri, & Khorrami, 1999). In GSA definition, the gravitational constant is set using the Eq. (6) (see Rashedi et al., 2009), where G_0 is the initial value, T the number of iterations and α a parameter.

$$G(t) = G_0 \exp\left(-\alpha\frac{t}{T}\right) \tag{6}$$

The total force $F_i^d(t)$ that acts on agent i in a dimension d is calculated by a random way (see Eq. (7), where $rand_j \in [0, 1]$).

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}^d(t) \tag{7}$$

On the base of (2), the acceleration of the agent i at time t, and in direction d^{th}, $a_i^d(t)$, is given by (8), with M_{ii} inertial mass of i^{th} agent.

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \tag{8}$$

The velocity v and the position x of an agent at $t+1$ time are calculated by (9) and (10) respectively, with Δt=1.

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t)\Delta t \tag{9}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1)\Delta t \tag{10}$$

The Eqs. (11)–(13) update the values of the masses according to the fitness values of agents $fit_i(t)$ for minimization or maximization, where fit_i is a function of the position X_i of the ith agent, i.e. $fit_i = f(X_i)$.

$$M_{ai} = M_{pi} = M_{ii} = M_i, i = 1, 2, \ldots, n \tag{11}$$

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{12}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{n} m_j(t)} \tag{13}$$

The quantities $best(t)$ and $worst(t)$ are defined by (14) and (15) for a minimization problem, whereas the Eqs. (16) and (17) refer to maximization problems.

$$best(t) = \min_{j\in 1,\ldots,n} fit_j(t) \tag{14}$$

$$worst(t) = \max_{j\in 1,\ldots,n} fit_j(t) \tag{15}$$

$$best(t) = \max_{j\in 1,\ldots,n} fit_j(t) \tag{16}$$

$$worst(t) = \min_{j\in 1,\ldots,n} fit_j(t) \tag{17}$$

Taking into account only the best agents which attract the others (K_{best}), the Eq. (7) is changed in (18).

$$F_i^d(t) = \sum_{j\in K_{best}, j\neq i}^{n} rand_j F_{ij}^d(t) \tag{18}$$

3. The Neuro and fuzzy Gravitational Search Algorithm

The main task is to assure a trade-off between exploration and exploitation by adjusting suitable parameters. First of all, we define the quantity denoted by P_p to measure the population progress (see (19))

$$P_p(i) = \left|\frac{fit_{mean}(i-1) - fit_{mean}(i)}{max(fit_{mean}(i-1), fit_{mean}(i))}\right| \tag{19}$$

where $fit_{mean}(i)$ is the fit mean value on all agents at the ith iteration with $i = 1, \ldots, N$, where N denotes the iterations number; note that $P_p \in [0, 1]$. The power of exploration and exploitation of GSA depends on the value of the gravitational constant G defined by (6). Note that, a small change of the parameter α in (6) provides an exponential change of G which in turn has a great effect in the acceleration of agents. Therefore, by controlling acceleration and velocity of agents, exploration and exploitation can be suitably tuned. So, we choose α as parameter to adjust. On the other hand, the number of agents which apply the force to other individuals in search space is defined by *Kbest* (see Eq. (18)). The parameter *Kbest* in Eq. (18) checks the number of agents that apply force to other individuals in search space. In GSA, *Kbest* decreases linearly in each iteration from n to 1, where n is the agents number. This means that at the end of iterations only one agent applies force to the other individuals. The idea is to reduce exponentially *Kbest*, so that the decrement of *Kbest* is more fast. Therefore, we define *Kbest* for each iterations i from 1 to N as in (20).

$$Kbest(i) = \left\lfloor N \exp\left(-\beta\frac{i}{N}\right)\right\rfloor \tag{20}$$

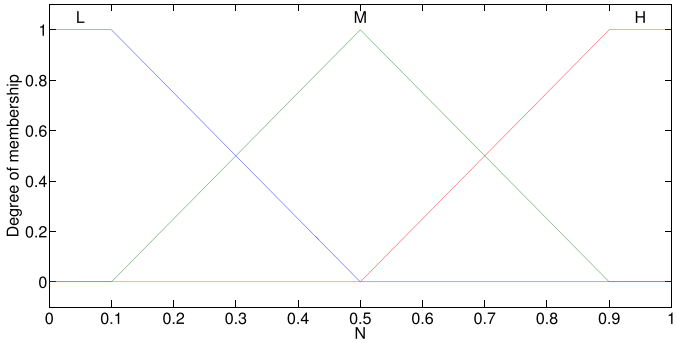

Fig. 1. Membership functions of the fuzzy input N.

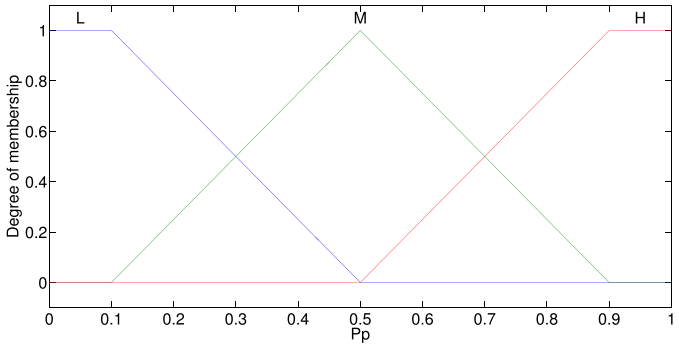

Fig. 2. Membership functions of the fuzzy input P_p.

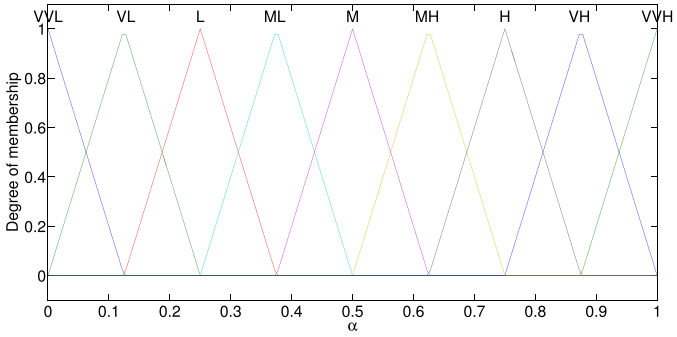

Fig. 3. Membership functions of the fuzzy output α.

Table 1
Fuzzy rules for the designed fuzzy system.

N	P_p	α
L	L	L
L	M	VL
L	H	VVL
M	L	MH
M	M	M
M	H	ML
H	L	VVH
H	M	VH
H	H	H

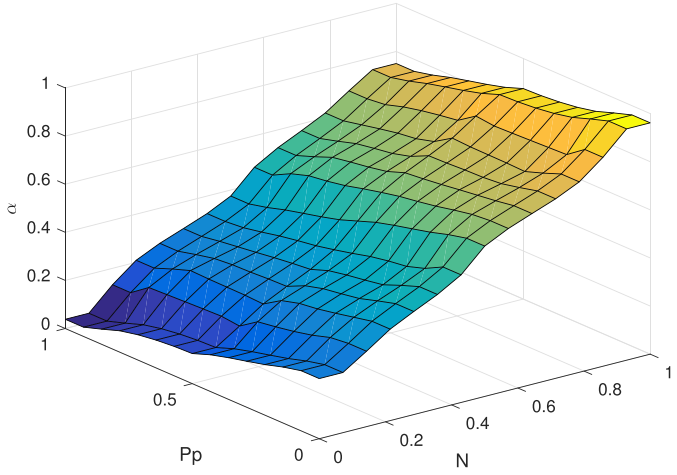

Fig. 4. Output surface of fuzzy system.

The value of the parameter β in (20) defines the decreasing rate of *Kbest*. The tuning of the parameter α must aim to improve exploration and exploitation capabilities of our algorithm. To achieve this goal, we decide of tuning α through a suitable Fuzzy Inference System (FIS). Note that, the value of the parameter α affects the gravitational constant value G (see (6)).

The main steps in FIS models development include the input/output variable identification, the choice of the membership functions (MF) type and number, and definition of rules.

The number of MFs per variable depends on the range and features of the variable. The accuracy of a FIS depends on number of MFs: the complexity of a fuzzy system increases with the membership functions number increment. However, MFs number affects the complexity of the system, therefore a good compromise between accuracy and complexity of FIS's have to be achieved. We consider two inputs: the iterations number N and the population progress P_p defined by Eq. (19). The fuzzy output of FIS-α has the task of adjusting the value of the parameter α in the Eq. (6). Each input has three MFs denoted as Low (L), Medium (M) and High (H). Moreover, to assure a good performance of FIS and to have more refined values of α, 9 MFs are defined: Very Very Low (VVL), Very Low (VL), Low (L), Medium Low (ML), Medium (M), Medium High (MH), High (H), Very High (VH), Very Very High (VVH). Because triangular/trapezoidal functions are observed to result in better performance of FIS for a generic system (Azizipanah-Abarghooee, Terzija, & Golestaneh, 2016), triangular/trapezoidal MF's are chosen. The Figs. 1 and 2 show MF's for the inputs N and Pp respectively, whereas Fig. 3 show MF's for the output α. All the fuzzy variables assume values in the range [0; 1]; in the proposed algorithm these values will be normalized.

The definition of fuzzy rules is the core of our approach. By means of an expert knowledge/intuitive analysis of GSA, the rule base is developed. The Table 1 shows the fuzzy rules for FIS-α. The rules take into account the following concepts. At the beginning of iterations, i.e. when $N = L$, to assure exploration the agents must have a big acceleration, therefore a low value of α is necessary. In fact, if α is low than G increases (see (6)) and by (3) and (7), it follows that the acceleration (Eq. (8)) tends to increase. On the other hand, an early lack of improvement, that is $P_p = H$ and $N = L$, is a sign of premature convergence: with a very very low value of α the individuals can escape from local optima. When GSA is at the middle of the procedure ($N = M$) and there is lack of improvement ($P_p = H$), the values of α must be basically low. At the end of the iterations ($N = H$), if there is an improvement in the optima research ($P_p = L$) than α must be very very high. High values of α tend to decrease the value of the gravitational constant and therefore the acceleration. The Fig. 5 shows the architecture of FIS-α to adjust the parameter α. The Fig. 4 shows the output surface of the designed fuzzy system.

In order to assure learning capability from data of our fuzzy system, a suitable Neural Network (NN) is needed. NN are characterized by different topologies and training techniques. A directed linked topology to synchronize coupled reaction-diffusion neural

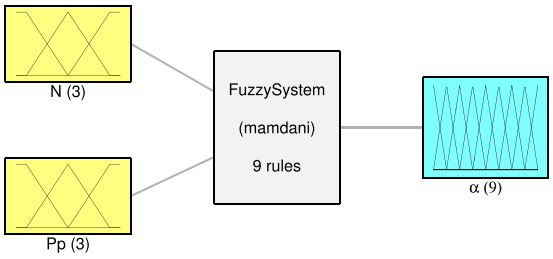

Fig. 5. Architecture of FIS to adjust α.

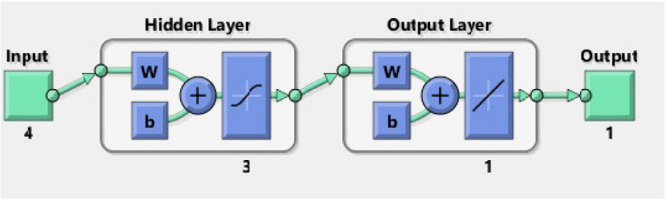

Fig. 6. Architecture of the designed neural network.

networks has been proposed by Zhang, Sheng, and Zeng (2017). Neuroevolution has been used to optimize the topology of NN (Vargas & Murata, 2017). Wen, Yu, Hu, Cao, and Yu (2015) studied the global pinning synchronization problem for a class of complex networks with switching directed topologies. Refer to the training methods, the Back-Propagation (BP) algorithm (Rumelhart, Hinton, & Williams, 1986) is a commonly used method in neural network learning systems. It involves minimization of an error function which is accomplished by performing gradient descent search on the error surface (Gallant, 1993). Hagan, Demuth, and Beale (1996) increased the training speed of BP through the Marquardt algorithm for nonlinear least squares. However, Extreme Learning Machines (ELM) (Huang, Zhu, & Siew, 2006) supplies a learning speed thousands of times faster than BP. Privacy-preserving protocols for both the BP and ELM algorithms have been proposed (Samet & Miri, 2012). The results show that the performance of the ELM protocol is higher than that of the BP. Moreover, ELM have been used as learning algorithms for CNN showing very good learning time reduction capabilities (Kim, Kim, Jang, & Minho Lee, 2017). An analysis on the simplest way to train ELM has been accomplished by Akusok, Bjork, Miche, and Lendasse (2015).

Our NN must guarantee the exploration and exploitation phases during the search process. The first step for the design of a NN is the training set definition. In order to achieve good exploration and exploitation, the parameter α must increase with the iterations number. Therefore, the desired values of α are low at the beginning of search phase and high at the end of the iterations. Following experimental ways, we define a fast increment of α on iterations number from 1 to 200 and in the range $800 - 1000$, whereas in the range $200 - 800$ α increases slowly. Thus the output of our NN is the parameter α. Because the value of α depends on iterations number N, a first input of NN is the iteration number. Moreover, the other inputs are: the mean of the fitness function $\overline{f}$ and agents acceleration $\overline{a}$, and the Euclidian distance between agents R. Finally, we design a NN with 4 inputs and 1 output. Once defined inputs and outputs of NN and its training set, the choice of hidden layer neurons number is needed. From trial and error procedures, we choose of adding one hidden layer composed by 3 neurons. Therefore, a $4 - 3 - 1$ NN is designed (see Fig. 6). The training technique applied to the proposed NN is the Levenberg-Marquardt algorithm (Hagan et al., 1996) on $230,000$ patterns.

Now we describe the new version of GSA with parameters adaptation through the designed neural network and fuzzy system.

Algorithm 1 (Neuro and Fuzzy Gravitational Search Algorithm).

S1. Initialize the position $X(i, j)$ $(i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, d)$ of the agents in the search space, randomly; remind that n is the number of agents and d is the dimension of test function. Let up and low be the extrema of search interval, X is computed by

$$X(i, j) = rand(i, j)(up - low) + low \tag{21}$$

for each agent i and dimension j; where $rand(i, j)$ is a function which generates $i \times j$ random numbers between 0 and 1.

S2. Fix the extrema value α_{inf} and α_{sup} of fuzzy output α and the initial value for the first two iterations $\alpha(1)$ and $\alpha(2)$.

S3. For each iteration from 1 to N, execute the steps from $S3.1$ to $S3.10$.

S3.1. Check the search space boundaries for agents according to the position X; agents that go out of the search space, are reinitialized randomly, i.e. for the ith-agent such that in the jth dimension $X(i, j) > up$ or $X(i, j) < low$, compute $X(i, j)$ by using (21).

S3.2. Evaluate the agents; compute the fitness fit_i of each agent by passing the position X to the test function and select the minimum (or maximum) fitness value among agents.

S3.3. Calculate the masses of each agent, i.e. Eqs. (11)–(15).

S3.4. Compute the gravitational constant G according to the value of $\alpha(i)$, with $i = 1, 2, \ldots, N$ (see Eq. (6)).

S3.5. Compute $Kbest$. It can be calculated in two different ways: linearly (as in GSA (Rashedi et~al. , 2009)) and exponentially (as defined by (20)). At this step, only one of these ways can be chosen. If the exponential case is selected, $Kbest$ is computed by (20), setting the value of β. Therefore, the new value of $Kbest$ is computed.

S3.6. According to $Kbest$, calculate the acceleration of each agent in gravitational field, see Eqs. (3)–(8) and (18).

S3.7. Compute the range of the fuzzy output α through the designed NN. The values of the current iteration number, fitness function mean $\overline{f}$, agents acceleration mean $\overline{a}$ and Euclidian distance between agents R, are passed as inputs to NN which gives the optimal value of α, denoted by α_{NN} for the current iteration. From this value, the extrema α_{inf} and α_{sup} are computed by

$$\alpha_{inf} = \alpha_{NN} - \epsilon \tag{22}$$

and

$$\alpha_{sup} = \alpha_{NN} + \epsilon \tag{23}$$

where $\epsilon \in]0, 1[$.

S3.8. Compute the population progress $P_p \in [0, 1]$ as defined in (19).

S3.9. The values of normalized iteration number i_n and population progress P_p are passed as inputs to fuzzy system which gives a value of α between 0 and 1, denoted by α_{out}. This value is normalized between α_{inf} and α_{sup} with the formula

$$\alpha = \alpha_{out} \cdot (\alpha_{sup} - \alpha_{inf}) + \alpha_{inf}$$

S3.10. Update the velocity v (see equation 9) and position X (see equation 10) of agents.

S4. Give in output the value of best solution in the last iteration.

4. Complexity analysis and experimental results

The designed FIS and NN of the proposed algorithm have been created by using Matlab FIS and Neural Networks toolboxes.

Because our approach adds the contribution of a fuzzy system and a neural network to GSA, a strict complexity analysis is necessary. On the other hand, to compare GSA with our approach, the complexity has to be the same. For this analysis, we take into account strict methodologies proposed by Pelusi and Tallini for the

design of balanced codes (Pelusi, Elmougy, Tallini, & Bose, 2015; Tallini et al., 2016).

The algorithms GSA, MGSA and NFGSA make use of random functions to generate random values. Because the random numbers can be generated by different algorithms and GSA, MGSA, NFGSA employ random functions with the same cost, the random function complexity is not shown.

In order to compute GSA, a time $T = O(nd)$ operations for each iteration is needed: the proposed algorithm must have the same complexity of GSA.

Because the Algorithm 1 uses a NN, a complexity analysis of the training algorithm is needed. During NN design phase, the choice of a suitable training model is very important in terms of complexity. Because the cost of the training phase depends on the training set dimensions, low complexity training algorithms should be used. As stated above, the designed NN is trained through the Levenberg–Marquardt algorithm which trains neural networks at a rate 10–100 times faster than the usual gradient descent backpropagation method. The complexity of the Levenberg–Marquardt algorithm is $O(m^3)$, with m patterns number (Dartmann, Zandi, & Ascheid, 2014). On the other hand, a training model based on ELM assures comparable or better performances than other models (Samet & Miri, 2012) at a lower computational cost. In fact, the computational complexity of ELM is $O(ml^2 + l^3)$ (Akusok et al., 2015; Iosifidis, Tefas, & Pitas, 2015; 2017), with l hidden layer neurons number. Therefore, our future challenge will focus on the design of a suitable ELM to train our NN.

Let n_{r_α} and n_{mf_α} be the rules and input membership functions number of FIS-α respectively. The inclusion of FIS-α to GSA takes time $S_1 = O(n_{r_\alpha} n_{mf_\alpha})$. Refer to NN, we consider the nodes equally distributed and assume constant costs per neuron. The complexity of NN depends on the number of multiplications needed to compute the activation of all neurons in the ith layer of the net. Thus NN takes time $S_2 = O(n_{l_1} n_{l_2} n_{l_3})$, where n_{l_1} is the neurons number of layer 1 (i.e. the input layer), n_{l_2} is the neurons number of layer 2 (i.e. the hidden layer) and n_{l_3} is the neurons number of layer 3 (i.e. the output layer). Because $n_{r_\alpha} = 9$ and $n_{mf_\alpha} = 2 \times 3 = 6$ and $n_{l_1} = 4$, $n_{l_2} = 3$ and $n_{l_3} = 1$, the complexity of Algorithm 1 is $S = O(n_{r_\alpha} n_{mf_\alpha}) + O(n_{l_1} n_{l_2} n_{l_3}) + O(nd) = O(nd)$=T. Therefore the complexity of GSA and NFGSA is the same. Remind that n is the agents number and d the space research dimension.

Now, we analyze the complexity of MGSA proposed by Li et al. (2016). MGSA uses the mutation operator and crossover operation of DE to generate new agents to increase the diversity of the population when GSA tends to be premature. Adding this features, the complexity of classical GSA increases. In fact, considering the cost for each iteration, the step 8 of MGSA (see Li et al., 2016) gives the biggest cost among steps $2 - 8$. In particular, the complexity of MGSA is $O((nd)^2)$. Therefore, the complexity of MGSA is significantly greater than NFGSA and GSA.

Once verified that GSA and Algorithm 1 have the same complexity, the task is to improve the performances of GSA by using our algorithm.

The Algorithm 1 is run on $N = 1000$ iterations for the benchmark functions in Tables 2, 3 and on $N = 500$ for functions in Table 4, the number of agents n is 50, the dimension d is 30 for the unimodal and multimodal test functions (see Tables 2 and 3), whereas for multimodal functions with fix dimension, the quantity d depends on specific function (see Table 4). The parameter *Kbest* defined in (20) is computed with $\beta = 5$.

At the first iteration of NFGSA, the value of α must be initialized. In GSA, the parameter α is equal to 20 (Rashedi et al., 2009) for all the iterations: in NFGSA the initial value of α is set to 10.31, in other terms $\alpha(1) = 10.31$. From the second iteration to the last one, NN gives in output the values of α. These values are used to compute the range of fuzzy output membership functions; in other terms α_{inf} and α_{sup} are calculated by (22) and (23) with $\epsilon = 0.5$. Thus FIS-α can estimate the sub-optimal values of α.

Table 2
Unimodal functions.

Unimodal functions	S
$F1(X) = \sum_{i=1}^{n} x_i^2$	$[-100; 100]^n$
$F2(X) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10; 10]^n$
$F3(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-100; 100]^n$
$F4(X) = \max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	$[-100; 100]^n$
$F5(X) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i)^2 + (x_i - 1)^2\right]$	$[-30; 30]^n$
$F6(X) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	$[-100; 100]^n$
$F7(X) = \sum_{i=1}^{n} i \cdot x_i^4 + random[0, 1)$	$[-1.28; 1.28]^n$

Table 3
Multimodal functions.

Multimodal functions	S
$F8(X) = \sum_{i=1}^{n} -x_i \sin\sqrt{\lvert x_i \rvert}$	$[-500; 500]^n$
$F9(X) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i + 10)\right]$	$[-5.12; 5.12]^n$
$F10(X) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) +$ $-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32; 32]^n$
$F11(X) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600; 600]^n$
$F12(X) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 +$ $+10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} +$ $+\sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	$[-50; 50]^n$
$F13(X) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_1 - 1)^2[1 +$ $+\sin^2(3\pi x_1 + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} +$ $+\sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50; 50]^n$

Table 4
Multimodal functions with fix dimension.

Multimodal functions with fix dimension	S
$F14(X) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	$[-65.53; 65.53]^2$
$F15(X) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	$[-5; 5]^4$
$F16(X) = 4x_1^2 + 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5; 5]^2$
$F17(X) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	$[-5; 10] \times [0; 15]$
$F18(X) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 +$ $+6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 +$ $+48x_2 - 36x_1x_2 + 27x_2^2)]$	$[-5; 5]^2$
$F19(X) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	$[0; 1]^3$
$F20(X) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	$[0; 1]^6$
$F21(X) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0; 10]^4$
$F22(X) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0; 10]^4$
$F23(X) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0; 10]^4$

The FIS surface of Fig. 4 would seem not very complicated and easily replaceable with a plane surface. In other words, we could use a plane surface in the Step 3.9 of the Algorithm 1 to calculate α. Taking into account the same expert knowledge/intuitive analysis of GSA used for the fuzzy rules definition, a suitable plane surface is defined. Following a mathematical approach, we define a linear relation between α, N and P_p (see (24))

$$\alpha = a \cdot N + b \cdot P_p + c \tag{24}$$

where $a, b, c \in \mathbb{R}$. Note that, this is the simplest choice of function type. From the analysis of GSA, it follows that at the beginning of iterations with a slow population progress, the value of α must be low. Because in such approach specific numerical values are necessary, this situation can be represented by the sequence $(N, P_p, \alpha) = (0, 0, 0.25)$. At the beginning of iterations and with a high population progress, α must be very low: the relative sequence is $(N, P_p, \alpha) = (0, 1, 0)$. At the end of the algorithm and with a low

Table 5
Comparison between GSA, MGSA, PSGSA and NFGSA over 30 runs, population size 50 and functions dimension $d = 30$ for test functions F1-F13.

	GSA (Rashedi et al., 2009)	MGSA (Li et al., 2016)	PSGSA		NFGSA	
	mean	mean	mean	std	mean	std
*F*1	7.30×10^{-11}	3.60×10^{-34}	1.04×10^{-02}	1.91×10^{-02}	7.51×10^{-23}	3.30×10^{-22}
*F*2	4.03×10^{-05}	1.35×10^{-17}	3.77×10^{-02}	6.43×10^{-02}	7.69×10^{-09}	3.14×10^{-08}
*F*3	$1.60 \times 10^{+02}$	1.20×10^{-33}	$2.72 \times 10^{+02}$	$9.06 \times 10^{+01}$	$9.13 \times 10^{+01}$	$4.40 \times 10^{+01}$
*F*4	3.70×10^{-06}	3.88×10^{-18}	1.67×10^{-02}	1.54×10^{-02}	3.42×10^{-08}	5.19×10^{-08}
*F*5	$2.52 \times 10^{+01}$	$2.61 \times 10^{+01}$	$5.53 \times 10^{+01}$	$6.22 \times 10^{+01}$	$2.50 \times 10^{+01}$	5.27×10^{-01}
*F*6	8.30×10^{-11}	4.53×10^{-29}	3.33×10^{-02}	1.83×10^{-01}	$0.00 \times 10^{+00}$	$0.00 \times 10^{+00}$
*F*7	$1.80 \times 10^{+02}$	2.28×10^{-05}	1.05×10^{-01}	6.30×10^{-02}	5.06×10^{-02}	1.24×10^{-02}
*F*8	$-2.80 \times 10^{+03}$	$-6.62 \times 10^{+03}$	$-3.39 \times 10^{+03}$	$7.19 \times 10^{+02}$	$-2.68 \times 10^{+03}$	$3.89 \times 10^{+02}$
*F*9	$1.53 \times 10^{+01}$	$9.95 \times 10^{+00}$	$2.13 \times 10^{+01}$	$5.34 \times 10^{+00}$	$2.33 \times 10^{+01}$	$5.35 \times 10^{+00}$
*F*10	6.90×10^{-06}	8.88×10^{-16}	1.48×10^{-02}	2.38×10^{-02}	5.16×10^{-12}	1.69×10^{-11}
*F*11	2.90×10^{-01}	1.08×10^{-14}	$1.01 \times 10^{+01}$	$2.47 \times 10^{+00}$	$1.50 \times 10^{+00}$	7.92×10^{-01}
*F*12	1.00×10^{-02}	4.34×10^{-31}	1.17×10^{-02}	2.98×10^{-02}	2.07×10^{-02}	5.20×10^{-02}
*F*13	3.20×10^{-32}	7.77×10^{-32}	1.30×10^{-01}	1.10×10^{-01}	1.10×10^{-03}	3.35×10^{-03}

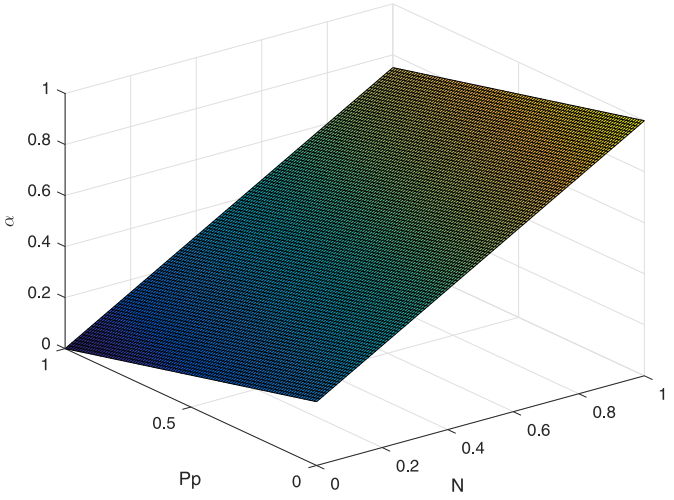

Fig. 7. Proposed plane surface.

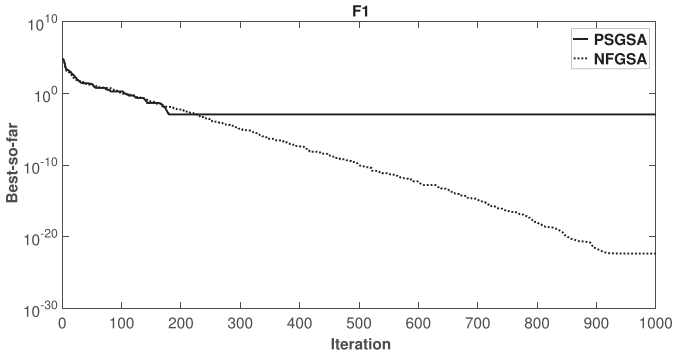

Fig. 8. Best-so-far solution trend for function *F*1.

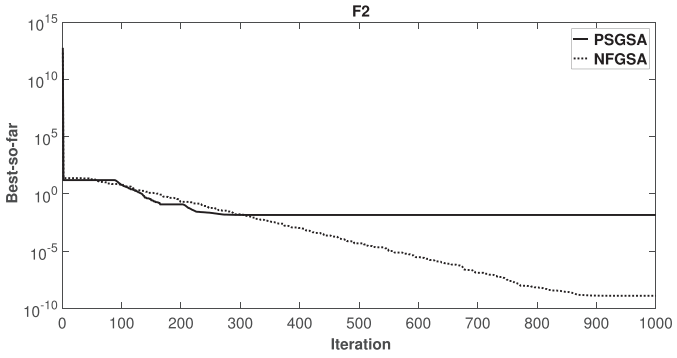

Fig. 9. Best-so-far solution trend for function *F*2.

population progress, α must be high, i.e. $(N, P_p, \alpha) = (1, 0, 1)$. Because the equation in (24) is the equation of a plane, the plane passing through the three points above has equation

$$\alpha = 0.75N - 0.25P_p + 0.25 \tag{25}$$

The plane of Eq. (25) is shown in Fig. 7. Note that, the value of α changes dynamically according to the values of N and P_p, therefore the fuzzy computation of α in Algorithm 1 can be replaced with the calculation of α as defined by (25). Removing the Step 3.7 from NFGSA and using (25) in the Step 3.9 to calculate α, we obtain a new algorithm referred as Plane Surface Gravitational Search Algorithm (PSGSA).

Now, some comparisons between GSA, PSGSA, MGSA and NFGSA are illustrated.

The results are averaged over 30 runs and the average best-so-far solution of GSA, PSGSA, MGSA and NFGSA in the last iteration is reported for test functions $F1 - F13$ in Table 5.

From the comparison between PSGSA and NFGSA, it follows that NFGSA is better than PSGSA for all the test functions. Note that, the improvement for the function *F*1 is of 21 order of magnitude. The Figs. 8 and 9 show the Best-so-far solution versus iterations number for PSGSA and NFGSA for the benchmark functions *F*1 and *F*2 respectively. From the observation of these figures, it follows that PSGSA suffers from premature convergence: this means that a plane surface does not assure of avoiding the trapping in local optima. This fact leads to fascinating challenges on the definition of more complex functions to calculate α for improving the fuzzy results.

Note that, with the same complexity, NFGSA improves GSA except for test function *F*13. For all the other test functions, NFGSA is better than GSA. In particular, *F*11 is improved of 12 order of magnitude. Moreover, there is a case (function *F*6) where NFGSA is better than MGSA and two cases (*F*5 and *F*8) with similar results. For all the other functions, MGSA achieves better results than NFGSA. However, the computational complexity of MGSA is $O((nd)^2)$, whereas NFGSA has complexity $O(nd)$.

The results for the benchmark functions $F14 - F23$ are averaged on 30 runs and with 500 iterations as in Rashedi et al. (2009). Because in Li et al. (2016) there are not experimental results on the functions $F14 - F23$, the comparison with MGSA is not accomplished. The Table 6 shows that NFGSA improves GSA for the functions $F15$, $F19 - F23$. For the other functions, GSA and NFGSA

Table 6
Comparison between GSA, PSGSA and NFGSA over 30 runs, 500 iterations for test functions F14-F23.

	GSA (Rashedi et al., 2009)	PSGSA		NFGSA	
	mean	mean	std	mean	std
*F*14	$3.70 \times 10^{+00}$	$5.93 \times 10^{+00}$	$2.98 \times 10^{+00}$	$4.32 \times 10^{+00}$	$3.23 \times 10^{+00}$
*F*15	8.00×10^{-03}	4.52×10^{-03}	2.38×10^{-03}	2.78×10^{-03}	3.41×10^{-03}
*F*16	$-1.03 \times 10^{+00}$	$-1.03 \times 10^{+00}$	1.78×10^{-03}	$-1.03 \times 10^{+00}$	6.52×10^{-16}
*F*17	3.98×10^{-01}	3.98×10^{-01}	$0.00 \times 10^{+00}$	3.98×10^{-01}	$0.00 \times 10^{+00}$
*F*18	$3.00 \times 10^{+00}$	$3.02 \times 10^{+00}$	3.79×10^{-02}	$3.00 \times 10^{+00}$	1.78×10^{-15}
*F*19	$-3.74 \times 10^{+00}$	$-3.86 \times 10^{+00}$	2.48×10^{-05}	$-3.86 \times 10^{+00}$	2.65×10^{-15}
*F*20	$-2.06 \times 10^{+00}$	$-3.32 \times 10^{+00}$	1.32×10^{-15}	$-3.32 \times 10^{+00}$	1.36×10^{-15}
*F*21	$-6.08 \times 10^{+00}$	$-6.76 \times 10^{+00}$	$3.72 \times 10^{+00}$	$-6.29 \times 10^{+00}$	$3.72 \times 10^{+00}$
*F*22	$-9.34 \times 10^{+00}$	$-1.02 \times 10^{+01}$	$1.22 \times 10^{+00}$	$-1.04 \times 10^{+01}$	9.90×10^{-16}
*F*23	$-9.46 \times 10^{+00}$	$-1.05 \times 10^{+01}$	2.06×10^{-15}	$-1.03 \times 10^{+01}$	$1.22 \times 10^{+00}$

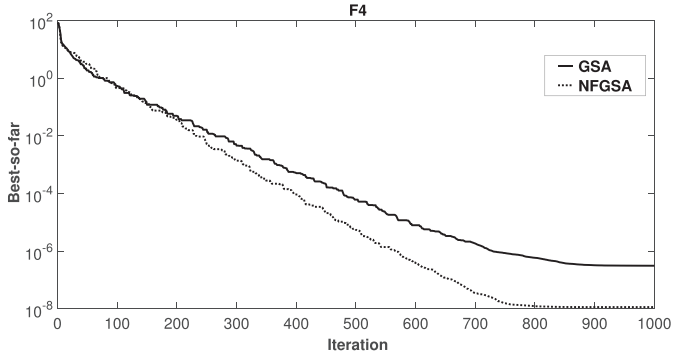

Fig. 10. Best-so-far solution trend for function *F*4.

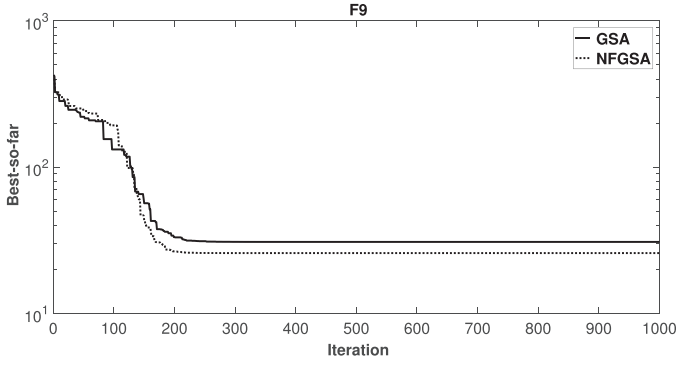

Fig. 12. Best-so-far solution trend for function *F*9.

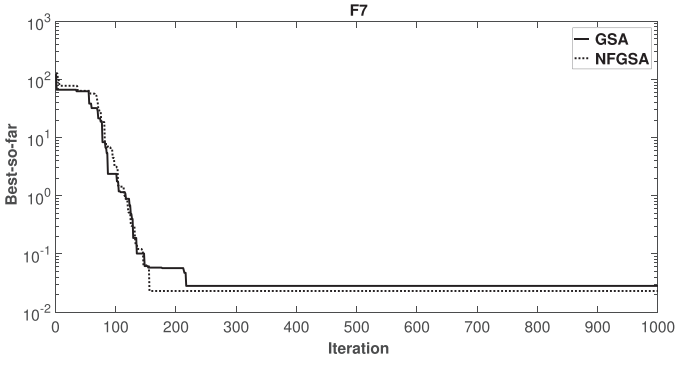

Fig. 11. Best-so-far solution trend for function *F*7.

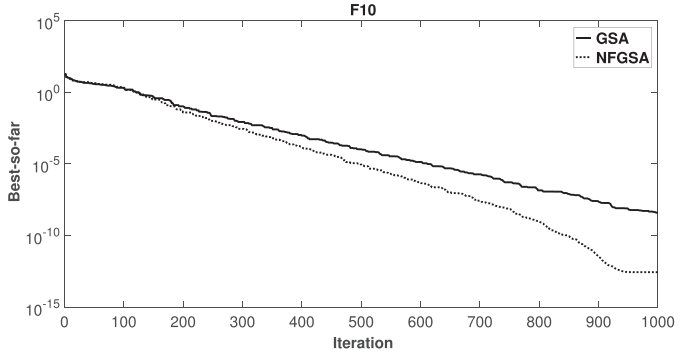

Fig. 13. Best-so-far solution trend for function *F*10.

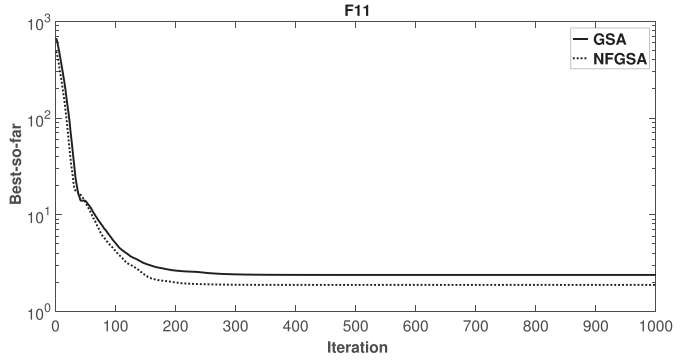

Fig. 14. Best-so-far solution trend for function *F*11.

achieve similar results. Moreover, the results of NFGSA are generally better than PSGSA.

The Fig. 10 shows the Best-so-far solution versus iterations number for GSA and NFGSA for the benchmark function *F*4. Note that NFGSA improves GSA because NFGSA tends to find the global optimum faster than GSA. A significant improvement there is for the function *F*7 (see Fig. 11) where the best-so-far solution curve of NFGSA is below GSA curve for the iterations from 150th to 1000th. Similar results are achieved for the function *F*9 (see Fig. 12). From the observation of the Figs. 13-15, we can note that NFGSA always has better convergence than GSA for the functions $F10 - F12$. In fact, the best-so-far solution curve of NFGSA is below GSA curve for all the iterations. This means that, NFGSA tends to find the global optimum faster than GSA and hence has a higher convergence rate. The Figs. 16–18 show that both the algorithms achieve the same optimum value, but NFGSA finds it before GSA.

In order to show the graphical charts significance degree, a statistical analysis of the results is necessary. Generally, the computation time of a search algorithm is a measure of its convergence speed. However, the computation time is not suitable for evaluating the convergence speed because it depends on the differences of the computer hardware performances, the operation system, the programming language, and so on. Therefore, for evaluating the convergence speed, it is a more objective criterion to take into account the number of function evaluations than the computation

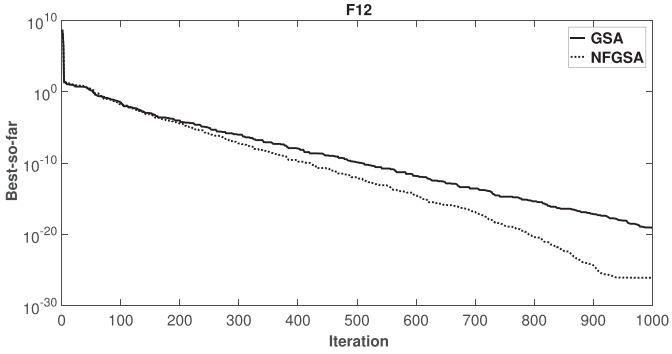

Fig. 15. Best-so-far solution trend for function *F*12.

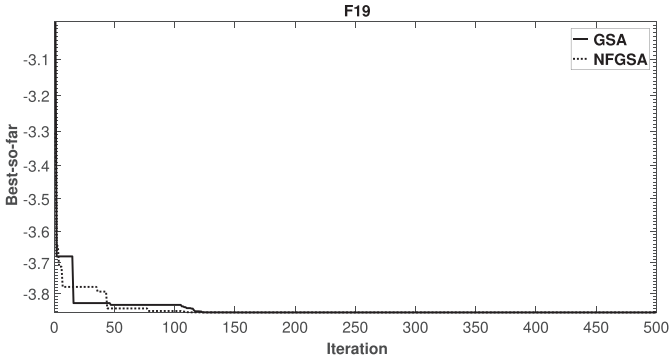

Fig. 16. Best-so-far solution trend for function *F*19.

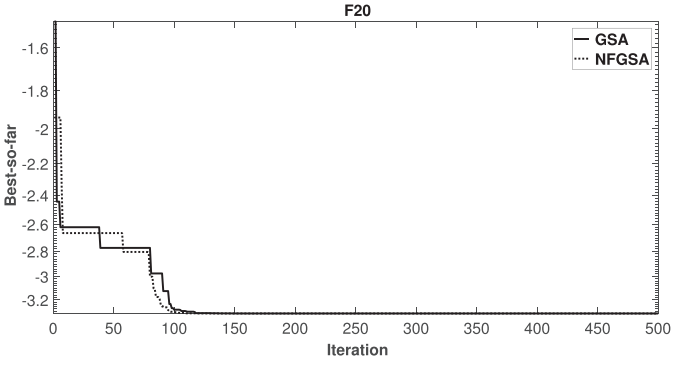

Fig. 17. Best-so-far solution trend for function *F*20.

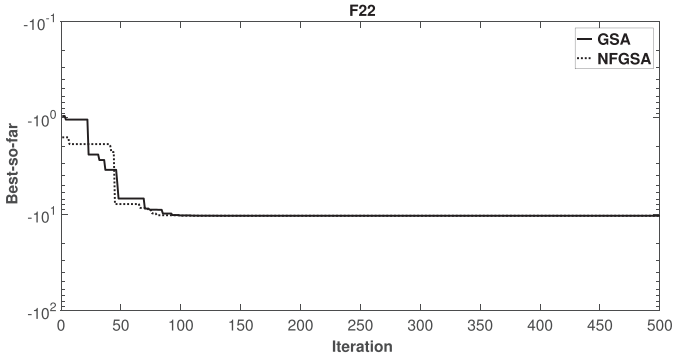

Fig. 18. Best-so-far solution trend for function *F*22.

Table 7
Convergence rate of GSA, PSGSA and NFGSA for the functions $F1 - F13$.

	GSA c_r		PSGSA c_r		NFGSA c_r	
	mean	std	mean	std	mean	std
*F*1	0.9967	0.0044	0.2402	0.1530	0.9956	0.0081
*F*2	0.9955	0.0039	0.3422	0.1830	0.9976	0.0035
*F*3	0.9990	0.0000	0.5398	0.0055	0.9641	0.0062
*F*4	0.9990	0.0000	0.5332	0.1270	0.9987	0.0008
*F*5	0.9990	0.0000	0.5550	0.0072	0.9655	0.0102
*F*6	0.0932	0.0061	0.0939	0.0105	0.0979	0.0042
*F*7	0.1717	0.0208	0.2443	0.1713	0.1683	0.0290
*F*8	0.0031	0.0025	0.0421	0.0979	0.0029	0.0025
*F*9	0.9274	0.0227	0.3696	0.0062	0.7212	0.0112
*F*10	0.9982	0.0013	0.2975	0.1594	0.9535	0.0084
*F*11	0.9990	0.0000	0.5209	0.0043	0.9309	0.0659
*F*12	0.9826	0.0409	0.4960	0.1522	0.9265	0.1110
*F*13	0.9965	0.0054	0.4992	0.1419	0.9613	0.0646

Table 8
Convergence rate of GSA, PSGSA and NFGSA for the functions $F14 - F23$.

	GSA c_r		PSGSA c_r		NFGSA c_r	
	mean	std	mean	std	mean	std
*F*14	0.0450	0.1388	0.0164	0.0223	0.0803	0.1714
*F*15	0.9342	0.2429	0.1074	0.0525	0.8903	0.2263
*F*16	0.8729	0.0655	0.1766	0.0794	0.7611	0.0582
*F*17	0.7326	0.0242	0.3335	0.0080	0.5965	0.0199
*F*18	0.9523	0.0493	0.2558	0.1139	0.7899	0.0693
*F*19	0.9068	0.0552	0.4078	0.0462	0.7715	0.0429
*F*20	0.9493	0.0155	0.3954	0.0370	0.7429	0.0114
*F*21	0.7340	0.3818	0.3791	0.1250	0.7133	0.1782
*F*22	0.9240	0.1588	0.4141	0.0833	0.7681	0.0497
*F*23	0.9297	0.1571	0.4381	0.0845	0.7860	0.2065

time. In particular, the convergence speed can be defined as the number of evaluations of the objective function until finding its minimum value (Precup, David, Petriu, Preitl, & Radac, 2014; Saha, Kar, Mandal, & Ghoshal, 2014; 2015; Tsai & Chou, 2006). Therefore, we define the convergence rate c_r as the ratio between the number of fitness function calculations n_f until finding its minimum value and the total number of evaluations, i.e. the iterations number N (see (26)).

$$c_r = \frac{n_f}{N} \tag{26}$$

The convergence rate is computed on 30 experiments. The Table 7 shows the convergence rate values of GSA, PSGSA and NFGSA for the test functions $F1 - F13$. The premature convergence of PSGSA is confirmed by the values of c_r in Table 7. The algorithm PSGSA remains trapped in local optima at the beginning of the search procedure or, at most, in the middle. Moreover, except for the functions *F*2 and *F*6, the convergence rate of NFGSA is better than GSA. The results of the convergence speed for the functions $F14 - F23$ are shown in Table 8. In this case, except for the function *F*14, the convergence speed of NFGSA is greater than GSA.

5. Conclusion

A neural network combined with a fuzzy system able to assure exploration and exploitation capability in a revised version of GSA have been designed. The novelties of the proposed approach are the new definition of the parameter *Kbest* of GSA and the design of suitable neural network and fuzzy system to adjust the parameter α of GSA. The neural network has been designed to supply optimal values of α to the fuzzy system which has the task of estimating sub-optimal values of α. Because the fuzzy system takes into account the population progress of the search phase, a suitable definition of the population progress has been proposed. In this way, a Neuro and Fuzzy Gravitational Search Algorithm has been designed. The performances of NFGSA are compared with GSA and

MGSA. The results show that, with the same computational complexity, NFGSA improves GSA in terms of global optimum search capability and convergence speed. Moreover, the proposed algorithm is better than MGSA for a test function and achieves the same results for other two functions. For all the other test functions, MGSA finds better values of global optimum than NFGSA. However, the computational complexity analysis shows that MGSA has complexity $O((nd)^2)$, whereas NFGSA has the same complexity of GSA, i.e. $O(nd)$. The comparison between NFGSA and PSGSA shows that NFGSA is better than PSGSA. Future research tasks will focus on the definition of an optimal function to calculate α and the design of a suitable ELM able to train the proposed NN.

Acknowledgment

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Akusok, A., Bjork, K. M., Miche, Y., & Lendasse, A. (2015). High-performance extreme learning machines: A complete toolbox for big data applications. *IEEE Access., 3*, 1011–1025.

Altinoz, O. T., Yilmaz, A. E., & Weber, G. (2014). Improvement of the gravitational search algorithm by means of low-discrepancy sobol quasi random-number sequence based initialization. *Advances in Electrical and Computer Engineering, 14*(3), 55–62.

Askari, H., & Zahiri, S. H. (2011). Data classification using fuzzy-GSA. In *Proceedings of the international econference on computer and knowledge engineering, mashhad, iran* (pp. 6–11).

Azizipanah-Abarghooee, R., Terzija, V., & Golestaneh, F. (2016). Multiobjective dynamic optimal power flow considering fuzzy-based smart utilization of mobile electric vehicles. *IEEE Transactions on Industrial Informatics, 12*(2), 503–514.

Castillo, O., Valdez, F., & Melin, P. (2007). Hierarchical genetic algorithms for topology optimization in fuzzy control systems. *International Journal of General Systems, 36*(5).

Clerc, M. (2006). *Particle swarm optimization*. Wiley-ISTE.

Dartmann, G., Zandi, E., & Ascheid, G. (2014). A modified levenbergmarquardt method for the bidirectional relay channel. *IEEE Transactions on Vehicular Technology., 63*(8), 4096–4101.

Doulamis, N. D., Doulamis, A. D., Panagakis, A., Dolkas, K., Varvarigou, T. A., & Varvarigos, E. (2004). A combined fuzzy-neural network model for non-linear prediction of 3-d rendering workload in grid computing. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 34*(2), 1235–1247.

Dowlatshahi, M. B., Nezamabadi-pour, H., & Mashinchi, M. (2014). A discrete gravitational search algorithm for solving combinatorial optimization problems. *Information Sciences, 258*, 94–107.

Eiben, A. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions Evolutionary Computer, 3*, 47–75.

Engelbrecht, A. P. (2005). *Fundamentals of computational swarm intelligence*. Wiley.

Fedorovici, L. O., Precup, R. E., Dragan, F., David, R. C., & Purcaru, C. (2012). Embedding gravitational search algorithms in convolutional neural networks for OCR applications. *IEEE International Symposium on Applied Computational Intelligence and Informatics*, 125–130.

Formato, R. A. (2007). Central force optimization: A new metaheuristic with applications in applied electromagnetics. *Progress in Electromagnetics Research, 77*, 425491.

Formato, R. A. (2008). Central force optimization: A new nature inspired computational framework for multidimensional search and optimization. *Studies in Computational Intelligence, 129*, 221–238.

Gallant, S. I. (1993). *Neural network learning and expert systems*. Cambridge, MA, USA: MIT Press.

Ghalambaz, M., Noghrehabadi, A. R., Behrang, M. A., Assareh, E., Ghanbarzadeh, A., & Hedayat, N. (2011). A hybrid neural network and gravitational search algorithm (HNNGSA) method to solve well known wessinger's equation. *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering, 5*(1), 147–151.

Gonzalez, B., Valdez, F., & Melin, P. (2016). A gravitational search algorithm using type-2 fuzzy logic for parameter adaptation. In *Nature-inspired design of hybrid intelligent systems, springer, volume 667, studies in computational intelligence* (pp. 127–138).

Gonzalez, B., Valdez, F., Melin, P., & Prado-Arechiga, G. (2015). Fuzzy logic in the gravitational search algorithm enhanced using fuzzy logic with dynamic alpha parameter value adaptation for the optimization of modular neural networks in echocardiogram recognition. *Applied Soft Computing, 37*, 245–254.

Gupta, A., Sharma, N., & Sharma, H. (2016a). Fitness based gravitational search algorithm. *IEEE international conference on computing communication and automation. noida. india.*

Gupta, A., Sharma, N., & Sharma, H. (2016b). Accelerative gravitational search algorithm. In *Advances in computing. communications and informatics. jaipur. india* (pp. 1902–1907).

Gupta, A., Sharma, N., & Sharma, H. (2017). Exploitative gravitational search algorithm. proceedings of sixth international conference on soft computing for problem solving. In *Advances in intelligent systems and computing, vol 546* (pp. 163–173). Springer.

Hagan, M. T., Demuth, H. B., & Beale, M. H. (1996). *Neural network design*. Boston, MA: PWS Publishing.

Hassoun, M. H. (1995). *Fundamentals of artificial neural networks*. MIT Press Cambridge.

Haykin, S. (1998). *Neural networks: A comprehensive foundation*. Prentice Hall.

Holland, J. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. USA: Bradford Books, Cambridge, MA.

Holliday, D., Resnick, R., & Walker, J. (1993). *Fundamentals of physics*. USA: John Wiley and Sons: Hoboken. NJ.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks, 4*, 251–257.

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing, 70*(13), 489501.

Iosifidis, A., Tefas, A., & Pitas, I. (2015). On the kernel extreme learning machine classifier. *Pattern Recognition Letters., 54*, 11–17.

Iosifidis, A., Tefas, A., & Pitas, I. (2017). Approximate kernel extreme learning machine for large scale data classification. *Neurocomputing, 219*(5), 210–220.

Kennedy, J., & Eberhart, R. C. (2001). *Swarm intelligence*. Morgan Kauffman.

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*(8), 1589–1597.

Kim, J., Kim, J., Jang, G. J., & Minho Lee, M. (2017). Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection. *Neural Networks, 87*, 109–121.

Kirkpatrick, S., Gelatto, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.

Krleza, D., & Fertalj, K. (2017). Graph matching using hierarchical fuzzy graph neural networks. *IEEE Transactions on Fuzzy Systems, 25*(4), 892–904.

Li, C., Chang, L., Huang, Z., Liu, Y., & Zhang, N. (2016). Parameter identification of a nonlinear model of hydraulic turbine governing system with an elastic toilet hammer based on a modified gravitational search algorithm. *Engineering Applications of Artificial Intelligence, 50*, 177–191.

Loganathan, A., & Kaliyaperumal, G. (2016). An adaptive HVS based video watermarking scheme for multiple watermarks using BAM neural networks and fuzzy inference system. *Expert Systems With Applications, 6*(3), 412434.

Mansouri, R., Nasseri, F., & Khorrami, M. (1999). 2007 effective time variation of g in a model universe with variable space dimension. *Physics Letters, 259*, 194–200.

Mendoza, O., Melin, P., & Licea, G. (2009). A hybrid approach for image recognition combining type-2 fuzzy logic, modular neural networks and the sugeno integral. *Information Sciences, 179*(13), 2078–2101.

Montiel, O., Castillo, O., Melin, P., Rodriguez Diaz, A., & Sepulveda, R. (2007). Human evolutionary model: A new approach to optimization. *Information Sciences, 177*(10), 2075–2098.

Olivas, F., Valdez, F., & Ostillo, O. (2016). Gravitational search algorithm with parameter adaptation through a fuzzy logic system. In *Nature-inspired design of hybrid intelligent systems. springer. volume 667. studies in computational intelligence* (pp. 391–405).

Ozan Kocadagli, O., & Langari, R. (2017). Classification of EEG signals for epileptic seizures using hybrid artificial neural networks based wavelet transforms and fuzzy relations. *Expert Systems With Applications*. doi:10.1016/j.eswa.2017.07.020.

Pelusi, D., Elmougy, S., Tallini, L. G., & Bose, B. (2015). M-ary balanced codes with parallel decoding. *IEEE Transactions on Information Theory, 61*(6), 3251–3264.

Pelusi, D., Mascella, R., & Tallini, L. (2017). Revised gravitational search algorithms based on evolutionary-fuzzy systems. *Algorithms, 10*(2), 44.

Peraza, C., Valdez, F., Garcia, M., Melin, P., & Castillo, O. (2016). A new fuzzy harmony search algorithm using fuzzy logic for dynamic parameter adaptation. *Algorithms., 9*(4), 69.

Precup, R. E., David, R. C., Petriu, E. M., Preitl, S., & Radac, M. B. (2014). Adaptive GSA-based optimal tuning of PI controlled servo systems with reduced process parametric sensitivity, robust stability and controller robustness. *IEEE Transactions on Cybernetics, 44*(11), 1997–2009.

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Science, 179*, 2232–2248.

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2010). BGSA: Binary gravitational search algorithm. *Natural Computing, 9*, 727–745.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536.

Saeidi-Khabisi, F. S., & Rashedi, E. (2012). Fuzzy gravitational search algorithm. In *Proceedings of the international e-conference on computer and knowledge engineering* (pp. 156–160).

Saha, S. K., Kar, R., Mandal, D., & Ghoshal, S. P. (2014). Gravitation search algorithm: Application to the optimal IIR filter design. *Journal of King Saud University Engineering Sciences, 26*, 69–81.

Saha, S. K., Kar, R., Mandal, D., & Ghoshal, S. P. (2015). Optimal IIR filter design using gravitational search algorithm with wavelet mutation. *Journal of King Saud University Computer and Information Sciences, 27*, 25–39.

Samet, S., & Miri, A. (2012). Privacy-preserving back-propagation and extreme learning machine algorithms. *Data & Knowledge Engineering, 7980*, 4061.

Schutz, B. (2003). *Gravity from the ground up*. UK: Cambridge University Press: Cambridge.
Setnes, M., & Roubos, H. (2000). GA-fuzzy modeling and classification: Complexity and performance. *IEEE Transaction Fuzzy Syst., 8*, 509–522.
Shi, Y., Eberhart, R., & Chen, Y. (1999). Implementation of evolutionary fuzzy systems. *IEEE Transaction Fuzzy Syst., 7*, 109–119.
Sombra, A., Valdez, F., Melin, P., & Castillo, O. (2013). A new gravitational search algorithm using fuzzy logic to parameter adaptation. In *Proceedings of the IEEE congress on evolutionary computation. cancun. mexico* (pp. 1068–1074).
Storn, R., & Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*, 341–359.
Tallini, L. G., Pelusi, D., Mascella, R., Pezza, L., Elmougy, S., & Bose, B. (2016). Efficient non-recursive design of second-order spectral-null codes. *IEEE Transactions on Information Theory., 62*(6), 3084–3102.
Tsai, J. T., & Chou, J. H. (2006). Design of optimal digital IIR filters by using an improved immune algorithm. *IEEE Transactions on Signal Processing, 54*(12), 4582–4596.
Vargas, D. V., & Murata, J. (2017). Spectrum-diverse neuroevolution with unified neural models. *IEEE Transactions on Neural Networks and Learning Systems, 28*(8), 1759–1773.
Wen, G., Yu, W., Hu, G., Cao, J., & Yu, X. (2015). Pinning synchronization of directed networks with switching topologies: A multiple lyapunov functions approach. *IEEE Transactions on Neural Networks and Learning Systems, 26*(12), 3239–3249.
Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, (1), 67–82.
Zhang, H., Sheng, Y., & Zeng, Z. (2017). Synchronization of coupled reaction-diffusion neural networks with directed topology via an adaptive approach. *IEEE Transactions on Neural Networks and Learning Systems*, 1–12.
Zhou, S., Chen, Q., & Wang, X. (2014). Fuzzy deep belief networks for semisupervised sentiment classification. *Neurocomputing, 131*, 312–322.