IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2020.DOI

Particle Swarm Optimization with Probability Sequence for Global Optimization

H. T. RAUF[1], U. SHOAIB[2], M. I. LALI[3] (Member, IEEE), M. Alhaisoni[4], M. N. Irfan[5], and M. A. Khan[6],

[1]Department of Computer Science, University of Gujrat, Gujrat, Pakistan. (e-mail: hafiztayyabrauf093@gmail.com)
[2]Department of Computer Science, University of Gujrat, Gujrat, Pakistan. (e-mail: umar.shoaib@uog.edu.pk)
[3]Department of Computer Science, University of Education, Lahore, Pakistan. (e-mail: ikramlali@gmail.com)
[4]College of Computer Science and Engineering, University of Ha'il, Ha'il, Saudi Arabia. (e-mail: majed.alhaisoni@gmail.com)
[5]Ryerson University, Toronto, Canada. (e-mail: naeem@ryerson.ca)
[6]Department of Computer Science, HITEC University Taxila. (e-mail: attique.khan@hitecuni.edu.pk)

Corresponding author: M. I. Lali (e-mail: ikramlali@gmail.com)

ABSTRACT Particle Swarm Optimization (PSO) has been frequently employed to solve diversified optimization problems. Choosing initial placement for population plays an important role in meta-heuristic methods since they can significantly converge. In this study, probability distribution has been introduced to enhance the diversity of swarm and convergence speed. Population initialization method based on uniform distribution is normally used when there is no preceding knowledge available regarding the candidate solution. In this paper, a new approach to initialize population is proposed using probability sequence Weibull marked as (WI-PSO) that applies the probability distribution to generate numbers at random locations for swarm initialization. The proposed method (WI-PSO) is tested on sixteen well-known uni-modal and multi-modal benchmark test functions broadly adopted by the research community and its encouraging performance is investigated and compared with the Exponential distribution based PSO (E-PSO), Beta distribution based PSO (BT-PSO), Gamma distribution based PSO (GA-PSO) and Log-normal distribution based PSO (LN-PSO). Artificial Neural Networks (ANNs) have become the most powerful tool for classification of complex benchmark problems. We have experimented the proposed method (WI-PSO) for weight optimization of a feed-forward neural network to ensure its purity and have compared with conventional back-propagation algorithm (BPA), E-PSO, BT-PSO, GA-PSO and LN-PSO. Due to flexible behaviour in the degree of freedom, the experimental results infer the perfection and dominance of the Weibull based population initialization. The result exhibits the anticipation of influence exerted by the proposed technique on all sixteen objective functions and eight real-world benchmark data sets.

INDEX TERMS Particle Swarm Optimization, Weibull Distribution, Neural Networks

I. INTRODUCTION

Optimization has been considered as an active area of research in the field of Artificial Intelligence (AI) for the past few years. Many real-world optimization problems are becoming more complex due to their non-linear behaviour of the complex structural design [1], [2]. To deal with these complex optimization problems, sophisticated and intelligent algorithms are required. The ultimate goal of the optimization is to reach the best possible solution [3].

Data classification has been an impressive research region for the past few decades [4], utilized for the partition of various occurrences of certain complex real-world problems. Data classification includes data imputations, analysis of medical diagnosis, simultaneous predictions, targeted marketing, weather forecasting, credit card fraud detection, value assessments, seasonal decomposition, and analysis of automatic abstraction [5]. Data classification paradigms provide successful search methods based on natural evolution and genetic process emerges in evolutionary computing [6]. These genetic processes govern the rules that are inspired by the natural behaviour of humans and insects to deal with global optimization problems [7]. The major reason for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3002725, IEEE Access

implementing evolutionary techniques on the vast amount of data to extract knowledge is the robustness and adaptive search-ability of global search methods [8].

Swarm Intelligence (SI), attribute to a kind of interpretive capability that appear in the communication of processing units [9]. The theory of swarm describes stochastic manner, multiplicity, messiness, and randomness where the theory of intelligence proves that the analytical ability is successful in some ways [10], [11]. A swarm composed of processing units, which can be computational or mechanical i.e. swarm of human beings and swarm of robots. In 1993 Beni [12] identified the term swarm intelligence (SI) in his research of cellular robotic systems. SI takes natural inspiration from the common behaviour of human beings as well as insects like termites, ants, wasps, bees and other social animal groups like the flock of birds and school of fish [13], [14]. An individual in the swarm can be described as a non-sophisticated solution; however, they have a strong ability to deal with complex non-linear problems cooperatively [15].

Kenndy and Eberhart carried out PSO in 1995 [16], for the training of neural networks and to solve non-linear optimization problems. Simple insights in PSO are the human cognition of natural behaviour like learning of human beings is dependent on their environmental behaviour; they talk with others and encode their pattern into their learning ways. This learning phenomenon has been embedded in PSO to solve optimization problems. Especially, in the divergent field of science PSO has become more natural to deal with non-linear complex optimization problems. The population of vector solution in PSO termed as a swarm and it is examining the new search spaces accumulating the evolution of school of fish while seeking for food. All individuals in the swarm known as particles transforming information and following best experience of each others and the previous best experience of their own to find global optimum [16]. Each particle must go through the basic rule of deciding the location of its own previous best or its neighbour.

The advents of global evolutionary methods such as PSO, ACO and other algorithms have determined the suitable approaches to handle complex optimization problems such as function minimization problems [17], [18], ANNs, expert systems and fuzzy systems, are the most promising applications of global optimization [19], [20]. In the recent decades, PSO has been widely implied to broaden the learning factors, weight minimization, and architecture of ANNs [21]–[23].

Random number generators are broadly emended in population-based methods for example genetic algorithm, PSO, and ACO for the purpose of population initialization [1]. The decision of selecting random number generator to initialize swarm may be useless for small uni-model and single dimensional problem. However, the selection of right distribution is compulsory when targeting complex multi-model non-linear optimization functions. The best population initialization increases the chance to reach maximum of fitness value at early stage. Similarly, selecting worst random number generator for the starting points of swarm may lead to stuck in local minima [24] and hence results in sub-optimal solutions. The reasonable fact for this indigent performance is the low coverage of search space with the particles. If the initialization is robust and particles cover all the search space with the random pattern that follow any distribution, then there is a less chance of total miss hits in terms of capturing global optimum [25]. This improvement in the performance of population-based PSO makes a great impact when solving the real-life classification problems or training of ANN classifiers.

There are three popular families of distribution for random number generation as pseudo-random sequences (uniform distribution), quasi-random sequences i.e. Sobol, Torus, Halton, Faure, Van der corput and the probability sequences i.e. Exponential, Beta, Gamma, Log-normal [26]. It has been proved that quasi-random sequences perform better as compared to pseudo-random sequences due to its low discrepancy where the performance of probability sequences is superior to the performance of quasi-random sequences [27]. For pseudorandom sequences the possible discrepancy can be described as $(Log(LogL))^{1/2}$ that is not enough lowest possible discrepancy to avoid local minima in PSO [28]. Quasi-random sequences have discrepancy of $(LogL)^{1/2}/L$, which tends to reach the lowest possible discrepancy. However, the probability sequences are the continuous probability sequence which depends on the shape and scale factor of particular distribution. This continuity in probabilistic models differentiates it from other low discrepancy sequence in terms of performance.

It is seen that, no adequate research has been completed to examine probability sequences to initialize swarm [27]. Keeping this perspective as target, we have carried out a new probability distribution based PSO initialization called Weibull distribution (WI-PSO). Weibull distribution follows continuous probabilistic model to compute the posterior of each number to form a sequence. Due to the patterned and robust dispersion rate for Weibull distribution that is computed using the proportional against the power of time, this distribution helps particles to cover the complete search space in a more patterned way. We have tested the proposed technique (WI-PSO) on several uni-model and multi-model bench mark functions listed in Table 2. For the performance comparison eight data sets have been taken from the famous machine learning repository UCI [29].

The rest of the paper is structured as, Section 2 presents related work and section 3 describes the overview study that includes ANNs and the training along with basic PSO working. The proposed methodology is discussed in section 4. Results are discussed in section 5 and the conclusion is provided in section 6.

II. RELATED WORK

Quasi-random sequences and probability distributions have been implemented by the researchers for the purpose of initializing swarm to enhance the performance of stochastic evolutionary algorithms. Comparison of the probability distribution with the quasi-random sequences has been carried out

by the authors in swarm [27]. The results of their experiments revealed that random number generator based on probability distribution shows better convergence rate as compared to quasi-random number generator and further in probability distribution family Exponential distribution based PSO (E-PSO) outperforms BT-PSO and Gaussian distribution based PSO G-PSO.

Halton sequence has been implied by the authors in [30] to reach the minimum of objective function. Halton sequence which falls under quasi-random sequences was tested on several multi model benchmark functions with the pseudo-random sequences following the uniform distribution. Halton sequences gave better results as compared to pseudo-random sequences. Zhang et al. [31] pioneered in a dual-environmental particle swarm optimizer that was able to deal with noisy and noise-free environments and tested real optimization problems with diverse population initialization.

The integrity of low-discrepancy sequence was verified over random number generator following the uniform distribution. The use of Faure and Sobol sequence can be observed in [32] for population initialization. A new quasi-random sequence is suggested by vander Corput [33] termed as van der Corput sequence which is associated with other low-discrepancy sequences. Van der Corput sequence is with base 2 series to generate random points. Krohling and Coelho [34] proposed improved version of PSO by adopting Exponential distribution (E-PSO) to initialize the swarm and compared it with uniform distribution on multi model test functions. Historical memory fitness for each individual is retained [35], to dynamically choose the next global best solution. The proposed approach helps in the exploration of particles for better optimization.

The performance of Exponential distribution based PSO (E-PSO) was better over random number generator. Many researchers have relied on probabilistic models of random number generator for mutation process of particles. Beta distribution has processed in adaptive mutation of PSO by the authors [36]. Adaptive mutation with beta distribution has shown promising results over simple version of PSO based on uniform distribution. Similarly, a Quasi-entropy index is employed by Cao et al. [37] to improve the local search performance of particles in the same direction based on entropy of each individual.

Many other autonomous simplex techniques or random number sequences other than probability distribution, low discrepancy sequences and pseudo-random utilized by the researchers for determining the origin spots of particles at vector search area. Non-liner simplicity (NSM) based method endorsed by Parsopoulos and Vrahatis for choosing initial placement for the particles [38]. In NSM based initialization, it covers the initial $E+1$ entries of the dimensions $D+1$. The first $V\,(D+1)$ particle goes through the process of NSM against each vertex V to find the initial points to be placed. NSM based initialization tends to increase the convergence speed of particles over simple initialization. An opposition-based hybrid initialization strategy is proposed by Kang et al. [39]. Their reconnected approach becomes highly effective in the noisy environment on the many objective optimization problems

Centroidal Voronoi Tessellations (CVT) generator was suggested by Mark and Ventura [40] to generate numbers at the locations identified by the CVT process. CVT is the process of the division of a search region into different unique cells. In the first K iteration each c_i cell is filled with the 1 to n_i particles. In the next $K+1$ iteration the particles n_{i+1} to n will be placed in c_{i+1} to n cells. Each CVT generator is correlated with one of the subset cell based on the point vicinity according to the assigned distance capacity. This distance capacity remains constant for the current K iterations up to $K+1$ to further assigns cells to the relevant particle. CVT method is analysed against uniform distribution based PSO and the mathematical results have conferred the CVT method robust in terms of improving swarm diversity. Neural networks have been adopted to dynamically select the new position for the particle in the entire search space [41]. Each hidden node in the network acts as objective function for local best particle.

The opposition-based initialization (O-PSO) mechanism was proposed [42]. Every materialistic thing contains two aspects of positivity and negativity similarly good nature and bad nature is a natural phenomenon of human beings. All humans cannot be all good or all bad at the same time. This natural behaviour of human beings is integrated into O-PSO to initialize the swarm. In O-PSO population is divided into two equal chunks, the first chunk is fired with the same direction into the search space where other chunk is fired with opposite direction of the search space. Under this concept, the entire population is characterized by same particles and opposite particles. Same particles can be describe as $p_t^s \in p, q$ where $p, q \in R$ and opposite particles are defined by $p_t^o = [p+q-p_t^s]\,; p, q \in R$ where R represents real number. O-PSO based initialization was examined under various multi-model benchmark function and tested with Simple PSO based uniform distribution the empirical results have proved that opposition-based initialization is sufficient for population initialization. Pareto active learning have been brought out by the authors in [43] for multi objective optimization problem. As compared to uni model or single objective optimization, selecting initial population is relatively more difficult.

Comparison of three different mechanism of initializing the swarm is carried out by Gutierrez et al. [44] .These techniques include PSO initialization chaotic distribution, the orthogonal array initialization, and opposition-based initialization. In orthogonal array initialization a collection of variables based on identical perturbation is calculated to initialize the swarm. Set of variables can be described as a matrix of size A*B with the orthogonal array of s. The chaotic distribution based PSO follows:

$$a_z = \beta.a_(z-1)(1-a_(z-1)) \tag{1}$$

Researchers have been involved to choose different starting configuration while dealing with real-world classification problem. The effectiveness of PSO on real-world classification

problem has been investigated [45]. They trained ANNs on 13 classical data set for the comparison of PSO effectiveness using the uniform distribution-based initialization. Similarly for the classification EMG signals Subasi [46] used Simple random number generator in PSO hybridized with support vector machines. Optimal Latin Hypercube Design OLHD based initialization method for population initialization is introduced by Zhao Liu [47] and analysed it with quasi-random sequence, Faure, Halton and Sobol on various real-life data mining problems. The above presented study is an evidence to reveal that selection of right starting configuration performs important job in terms of reinforce the swarm diversity and convergence rate. The non separable data is target by Xu et al. [48], the proposed study enhance the dynamic behaviour during the algorithm life cycle through intra-cluster distance (ICD) function.

III. NEURAL NETWORKS FOR CLASSIFICATION

A. NEURAL NETWORKS

ANNs have been employed since couple of decades to solve real-world complex problems [49]. ANNs have a processing unit that consists of neurons as ANNs take biological inspiration from the brain of human. ANNs are linked like a brain of the human with the interconnected neurons that are robust to the output of other connections that have changeable measures [50].

As ANNs have the ability of generalizing, adopting and learning the big amount of data therefore it has become the most powerful tool for classification of complex benchmark problems. The influence of ANNs over traditional classification models lies under the following facets. First, ANNs are enough flexible in that they can adjust themselves in the absence of distributional and functional form for the hidden layer model. Second they approximate any non-liner complex function with the hidden dimensions [19], [51]. One of the most widely used structures of ANNs is a feed-forward neural network. In this study, we have adopted feed-forward architecture of ANNs to classify real-world benchmark problems using the Weibull distribution based PSO (WI-PSO). In the feed forward network, there are three basic layers input, middle (sandwich) and output. Data frame is accepted through the input layer, middle layer acts as intermediate to carry the data farm while the interpretations of the model come through the output layer [52].

B. TRAINING OF THE NEURAL NETWORKS

The core objective is to train the feed-forward neural network to reach the optimized weights for enhancing the performance of the classifier. During weight optimization process of training the classifier, the model gets stuck in local minima which are most considerable problem for researchers [53]. The classical technique used for the training of intelligent classifier is back-propagation, which is an improved version of gradient decent method [54].

Gradient decent method is used to minimize the error function which was most of the time appear as Mean Square Error (MSE). In case of multi-modality and dependency of parameters, the performance of back-propagation method is low as it captures the local minima during the training phase. In this paper, we also have compared the performance of back-propagation method with the weibull distribution based PSO (WI-PSO).

C. PARTICLE SWARM OPTIMIZATION

PSO is a major contribution in the category of naturally inspired population-based techniques for decoding complex optimization problems. The structure of PSO is based on the composite practice of social insects like bees, ants, and a crowd of birds. The swarm consists of individuals, seeking for food in the entire search space; this can be translated into the population which consists of particles looking for an optimal solution in the multi-dimensional search area.

Due to the natural phenomena of inspecting food, particles have strong ability to explore new search areas for optimum candidate solutions. Every single individual inside the swarm is capable to reach the desired goal. To reach new best placement, each particle follows its previous best experience and the experience of its neighbour. Individual optimal solution is presented through the fitness value given by the particle.

PSO begins with the initialization of n particles in d dimensional search space. The current location for the ith particle during kth iteration is described by the $x_i = \{x_1, x_2, \ldots\ldots, x_n\}$. The previous best placement memorized by each individual is represented by $p_i = \{p_1, p_2, \ldots\ldots, p_n\}$ and the velocity, recently gained is $v_i = \{v_1, v_2, \ldots\ldots, v_n\}$. The vector P, designated to ${}^{best}_{i}p$, if its calculate the local best fitness of the entire group and represented by g_i^{best},if it contains the global best fitness of the entire swarm. The following two equations (Eq 2, Eq 3) control the functioning of basic PSO.

$$v_{i+1}=v_i+\tau_1\times\left({}^{best}_{i}p-x_i\right)+\tau_2\times\left({}^{best}_{i}g-x_i\right) \tag{2}$$

$$x_{i+1}=x_i+v_{i+1} \tag{3}$$

Here v_i represents the current velocity of particle and x_i shows current point where ith particle is placed. ${}^{best}_{i}p$ and ${}^{best}_{i}g$ are two vectors directing particle's local best solution and global best solution respectively at d dimensions in kth iteration. Where $\tau_1=(\mathrm{c}_1*\mathrm{Rand1}())$ and $\tau_2=(\mathrm{c}_2*\mathrm{Rand2}())$, c_1 and c_2 governs the acceleration impact on the particles. $\mathrm{Rand1}()$ and $\mathrm{Rand2}()$ are two random number generator following uniform distribution. The expression " $\times$ " is a vector multiplication. v_{i+1} and x_{i+1} defining the updated velocity and position vector of the particles in the entire swarm. Eq 2 is divided into three major blocks. v_i the momentum block representing old velocity of the particle, that indicates a specified distance have already covered by the particle, $\tau_1\times\left({}^{best}_{i}p-x_i\right)$ the cognitive block that returns the local best solution and $\tau_2\times\left({}^{best}_{i}g-x_i\right)$ the social block that reflects global best solution followed by the best of neighbouring particles.

IV. METHODOLOGY

Probabilistic models are reliable for population initialization than other low-discrepancy models and pseudo-random techniques. However, it depends on nature and dimensionality of the problem to select the population initialization method as uniform distribution is used where the parameters are independent of random numbers. In our proposed method PSO utilized random function rWeibull() to initialize the population. The pseudo code for the proposed method is presented in algorithm 1.

Algorithm 1 Proposed PSO

Require: $p_i = \{p_1, p_2, \ldots\ldots, p_n\}$ particles with the initial placement in the Swarm.

Ensure: $p_i = \{p_1, p_2, \ldots\ldots, p_n\}$ particles with the global best placement in the Swarm.

1: **for** each particle $p_i = \{p_1, p_2, \ldots\ldots, p_n\}$ at iteration k_1 **do**
2: Assign Initial values to the particles using Weibull generator as:

$$x_i = rWeibull();$$

3: Update ${}_i^{best}p_,$ if it gets better fitness score than x_i
4: Initialize $\mathrm{v_i}$ randomly.
5: Calculate optimal best solution ${}_i^{best}g_,$ from all of ${}_i^{best}p_,$.
6: Repeat the process up to k_n.
7: **for** each particle $p_i = \{p_1, p_2, \ldots\ldots, p_n\}$ **do**
8: Compute $\mathrm{v_{i+1}}$ from Eq 2.
9: Compute $\mathrm{x_{i+1}}$ from Eq 3.
10: **if** $\mathrm{x_{i+1}}$ gets better fitness score than ${}_i^{best}p_,$ **then**
11: Replace $\mathrm{x_{i+1}}$ to ${}_i^{best}p_,$
12: **end if**
13: **if** the fitness score of $\mathrm{x_{i+1}}$ is better than ${}_i^{best}g$ **then**
14: Update $\mathrm{x_{i+1}}$ with ${}_i^{best}g$.
15: Return $p_i = \{p_1, p_2, \ldots\ldots, p_n\}$ particles having global optimal solution.
16: **end if**
17: **end for**
18: **end for**

A. EXPONENTIAL DISTRIBUTION

In the field of probability theory, exponential distribution characterizes a time between the process of a random mathematical object (poisson point process) [55]. Poisson point process contains an independent and continuous event that occurs with constant average rate. The exponential distribution is memory less distribution. Memory less is a term used to define the object as time independent. Exponential distribution can be characterized by the probability density function which is given as:

$$f\left(a \mid \delta\right) = \begin{cases} \delta e^{-\delta a} & a \geq 0 \\ 0 & a < 0 \end{cases} \tag{4}$$

Where $f\left(a \mid \delta\right)$ can be defined by the function of Heaviside step for (a) .

$$f\left(a \mid \delta\right) = \begin{cases} \delta e^{-\delta a} * H(a) & a \geq 0 \\ 0 & a < 0 \end{cases} \tag{5}$$

The rate parameter is denoted by δ which controls the distribution over the interval of $\{0 \mid \infty\}$.

The cumulative distribution functions for the exponential distribution are presented in Eq 6.

$$F\left(a \mid \delta\right) = \begin{cases} 1 - e^{-\delta a} & a \geq 0 \\ 0 & a < 0 \end{cases} \tag{6}$$

Alternatively, $f\left(a \mid \delta\right)$ for the cumulative distribution can be described as in Eq 7.

$$F\left(a \mid \delta\right) = \begin{cases} 1 - e^{-\delta a} H(a) & a \geq 0 \\ 0 & a < 0 \end{cases} \tag{7}$$

The exponential function $rExponential()$ produces the random number over the interval [0, 1] to initialize the swarm which can be noted from Fig 1.

B. BETA DISTRIBUTION

Beta distribution is controlled by the two-shape parameters x and y over the interval of $0, 1$ where x = shape1 and y = shape2 represent the exponential components of random parameters [56]. In a broad class of disciplines, Beta distribution has been implemented to fit the model into random variables of finite length. This probability distribution has been successfully used in the Bayesian theorem where posterior probability and maximum likelihood is required to be computed. In case of Beta distribution, the probability density function is computed using the power function of factor t and its absorption $(1-t)$ where $1 \leq t \leq$ and $x, y > 0$. The power function is given below.

$$f\left(t \mid x, y\right) = R_{constant} \times t^{x-1}\left(1 - t^{y-1}\right) \tag{8}$$

In Eq 8 $R_{constant}$ is a random constant number which controls the realization factor of Beta distribution. Cumulative distribution function for the Beta distribution is given in Eq 9.

$$F\left(t \mid x, y\right) = \frac{\partial\left(t \mid x, y\right)}{\partial(x, y)} I_t \tag{9}$$

Where $\partial\left(t \mid x, y\right)$ represents the incomplete Beta function and I_t shows the function of improved version of regularized incomplete Beta. The particles use the function $rBeta()$ to get their initial places followed by the beta distribution pattern. Density Plot for Beta distribution is shown in Fig 2.

C. GAMMA DISTRIBUTION

Gamma distribution is managed by three continuous parameter measurements of random variable to form a random series. The first measure contains a scale factor $\emptyset$ and shape factor s, where the second measure contains a scale factor of $\alpha = \frac{1}{\emptyset}$ and shape factor $\beta = s$. Third measurement consists of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3002725, IEEE Access

average mean parameter $\rho = s/\alpha$ and the shape parameter s [57]. The Gamma distribution follows the rule of maximum entropy for measuring all the parameters which must be the real number and positive in nature. For a random variable r the Gamma distribution is maximum entropy according to $1/r$ base factor and uniform base factor. Shape parameter is also called rate parameter. We can characterize a random variable r in terms of rate factor $\alpha = \frac{1}{\emptyset}$ and the shape factor $\beta = s$ as shown in the Eq 10.

$$r \sim (\beta, \alpha) \equiv G(\beta, \alpha) \tag{10}$$

The probability density function for the Gamma distribution is shown in Eq 11.

$$f\left(r \mid \beta, \alpha\right) = \frac{\alpha^{\beta} t^{\beta-1} \varepsilon^{-\alpha t}}{(\beta)} \tag{11}$$

Where β represents a Gamma function.

The random variable r can also be described by rate factor $\emptyset$ and the shape factor s as shown in the Eq 12. The probability density function is presented in Eq 13.

$$r \sim (s, \emptyset) \equiv G(s, \emptyset) \tag{12}$$

$$f\left(r \mid s, \emptyset\right) = \frac{t^{s-1} \varepsilon^{-\frac{t}{\emptyset}}}{\emptyset^{s}(s)} \tag{13}$$

To occupy the space in the multidimensional search space particles adopt $rGamma()$ function to generate random points following the gamma distribution, which is shown in Fig 3.

D. LOGNORMAL DISTRIBUTION

A Lognormal distribution for a random variable l having a normally distributed log like $m = \ln(l)$ is normally distributed if l is Log normally distributed [58]. Similarly, $\exp(m)$ is log normally distributed for l, m if l is normally distributed. Log-normal distribution can be defined as: it is a vector product of multiplicative random variables followed by the central limit theorem. The probability density function and cumulative distribution function are given in Eq 14 and Eq 15, respectively.

$$f_L\left(m\right) = \frac{d}{dm} \Pr(\ln(M) \leq \ln(m)) \tag{14}$$

$$F_L\left(m\right) = \varphi(\frac{\ln\left(m\right) - \mu}{\tau}) \tag{15}$$

Where φ represents cumulative density and τ is cumulative probability. The $rLognormal()$ function has been employed to throw the particles into the search regions initially. Density Plot for Log-normal distribution is given in Fig 4.

E. WEIBULL DISTRIBUTION

Weibull distribution was proposed by the authors of [59] and it was implemented by Rosin and Rammler [60] in 1933 for the purpose of describing particle size. It is often considered as the exceptional case of Gamma distribution. The probability density function for the Weibull distribution is given in Eq 16.

$$f\left(w \mid \alpha, z\right) = \begin{cases} \frac{z}{\alpha}\left(\frac{w}{\alpha}\right)^{z-1} * e^{\left(-\left(\frac{w}{\alpha}\right)\right)^{z}} & w \geq 0 \\ 0 & w < 0 \end{cases} \tag{16}$$

Where $z > 0$ describes the shape factor and $\alpha > 0$ is the scale factor. The failure rate for Weibull distribution is computed using the proportional against power of time. Shape factor z is directly involved in this regard and can be interpreted as:

1) If $(z > 1)$, the relation between failure rate and time will be directly proportional.
2) If $(z < 1)$, time and failure will be inversely proportional.
3) If $(z = 1)$, shows the relation of failure rate over time.

For Alternative factor design the scale factor is changed to $\beta = \alpha^{-z}$. The hazard function is defined in Eq 17 and Eq 18 shows probability density function alternative factorization.

$$h\left(w; z, \beta\right) = \beta z w^{z-1} \tag{17}$$

$$f_L\left(w; z, \beta\right) = \beta z w^{z-1} * \exp(-\beta w^{z}) \tag{18}$$

Cumulative function of Weibull distribution is given in Eq 19.

$$F\left(w \mid z, \alpha\right) = 1 - e^{\left(-\left(\frac{w}{\alpha}\right)\right)^{z}} \tag{19}$$

The particles x_i are initialized randomly with Weibull distribution as: $x_i = rWeibull()$. The density Plot for Weibull distribution is presented in Fig 5.

V. RESULTS AND DISCUSSION

The proposed technique WI-PSO is simulated in C++ and implemented on a machine with 1.70 GHz Core (TM) i3-4010U CPU processor. A collection of benchmark test functions has been employed for comparison of proposed WI-PSO with other probability distribution-based E-PSO, BT-PSO, GA-PSO and LN-PSO for the purpose of measuring the performance of WI-PSO.

A. BENCHMARK FUNCTIONS

This section includes sixteen benchmark test function employed to examine the performance of the proposed technique. Table 2 contains the index of benchmark test functions where the search space interval represented by S and f_{min} shows the minimum value of global optimum. All of the individuals in the search space have object value that is assessed by the fitness function to be optimized in terms of minimizing the global fitness value. The average global fitness values for each fitness function are presented in the Table 3. The boundaries

for initialization of each of the distribution is defined in the Table 2 in terms of search space range.

B. PARAMETER SETTING

The initial parameter setting for simulation is structured as: the population size is 40 where the values for inertia weight is set to $w_{min} = 0.4$ and $w_{max} = 0.9$ over the interval of $[0.4, 0.9]$ and $c_1 = c_2 = 1.45$. The empirical results have been calculated for $D = 10$, $max_{iter} = 1000$; $D = 20$, $max_{iter} = 2000$ and $D = 30$, $max_{iter} = 3000$. In order to compare the performance of each method, the experiments were performed on 10 runs for each algorithm. The results are shown in Table 3 .

C. ANALYSIS

A comparison of proposed algorithm WI-PSO has been carried out with other existing algorithms E-PSO, BT-PSO, GA-PSO, and LN-PSO. The results of the conducted experiments are shown in Table 3. The results exhibit that the convergence rate of proposed WI-PSO is greater than other conventional techniques E-PSO, BT-PSO, GA-PSO and LN-PSO for 16 well-known benchmark functions. Excellent result for each test function is highlighted in bold.

From Table 3, we can see the fast convergence rate of the proposed method WI-PSO towards minimum of the test function. The results presented in Table 5 show that WI-PSO outperforms other techniques on eight real-world bench mark problems, which is a forecasting the use of WI-PSO for the purpose of population initialization. The convergence curves for all techniques can be found in Fig 7 to 16. We can see in the Fig 8 and 10 the proposed variant reaches to its minimum objective value 0 and same for the Fig 14 else all functions completed their 1000 generations. Surface and scatter Plot comparison of weibull to PSO in terms of Exploration vs Exploitation is given in the Fig 17 and 18. A comparative analysis in terms of pros and cons for all compared algorithm is presented in Table 1.

From Fig 7 to 16 the graphical representation of some well-known bench mark functions have been presented. It is clearly seen that the convergence rate of proposed method WI-PSO is better than other existing techniques E-PSO, BT-PSO, GA-PSO and LN-PSO.

D. DATA CLASSIFICATION EXPERIMENT

We have conducted experiments using eight benchmark datasets (Horse, Blood Tissue, Seed, Iris, Heart, Wine, Diabetes and Vertebral) taken from UCI repository. The researchers frequently use the datasets from this repository for evaluation of learning algorithms. Apart from this, the other reason for choosing these datasets was that traditional methods have examined them with variable completion. Training weights initialization has fixed randomly over the interval of $[-50, 50]$. Root Mean Squared Error (RMSE) metrics are adopted to record the accuracy. Table 4 shows the description of data set and Table 5 presents accuracy results.

1) Discussion

Training of multi-layer feed forward neural network was performed using the back-propagation algorithm (BPA), PSO based on Exponential distribution (E-PSO), PSO based on Beta distribution (BT-PSO), PSO based on Gamma (GA-PSO) distribution, PSO based on Lognormal distribution (LN-PSO) and the proposed algorithm Weibull distribution based PSO (WI-PSO). Eight real-world classification data set were selected to perform the comparison of all techniques. To evaluate the performance of the proposed technique properly, we have used k-fold cross validation. The value of folds k is fixed to $K = 10$. Each of the data set listed in Table 4 distributed into 10 equal folds for training of ANNs classifier. The ratio for testing and training is set to $1/9$ as 9 folds were used for training and 1 fold was used for testing. The experimental results of proposed method WI-PSO against other existing methods E-PSO, BT-PSO, GA-PSO and LN-PSO from Table 5 show the better accuracy as compared to other traditional approaches. Winning counts of proposed method from other techniques on all 8 bench mark data sets is 100%. WI-PSO can be effective choice for statistical problems as well as real-world classification problems. Fig 6 contains the accuracy comparison of the proposed technique with other traditional techniques.

VI. CONCLUSION

This paper presents a new method to choose a staring configuration in terms of population initialization. To verify the proposed technique, the comprehensive collection of standard uni-model and multi-model benchmark functions are employed. The experimental result reveals that the proposed techniques impressively sustain the heterogeneity of the swarm, enhances the convergence rate and discovers the considerable region of the swarm. The demonstration of proposed technique is to drive with a robust vector candidate solution if the previous information of the candidate solution is not available. The results explicitly tell that WI-PSO outperforms E-PSO, BT-PSO, GA-PSO, and LN-PSO. The performance of WI-PSO is better than E-PSO, BT-PSO, GA-PSO, and LN-PSO for training with the ANN classifier. The main purpose of this research is general but appropriate to other stochastic based meta-heuristic algorithms that promote the ultimate objective of our work. Regarding the application of population initialization, an extension for the near future is the hybridization of PSO with other evolutionary algorithms i.e, Simulated Annealing or Genetic Algorithms that could offer excellent results.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3002725, IEEE Access

TABLE 1: Comparative analysis in terms of pros and cons for all compared algorithm

	Pros	**Cons**
E-PSO	1. Better convergence rate 2. Significant for uni-model optimization	1. Low swarm diversity 2. High mean loss 3. Poor reliability analysis 4. Poor performance for real world applications 5. Stuck in local minima
BT-PSO	1. Better exploitation 2.Significant for uni-model functions 3. Better convergence rate 4. Low mean loss	1. Lwo swarm diversity 2. Not suitable for real world applications. 3. Poor reliability analysis 4. Stuck in local minima
GA-PSO	1.Better exploration 2. Low mean loss 3.Skewed and positive particle initialization	1. Not suitable for opposition based initialization 2. Low swarm diversity 3. Stuck in local minima
LN-PSO	1.Significant for log normal optimization functions 2. Better convergence rate	1. High mean error 2. Stuck in local minima 3. Poor swarm diversity 4. Poor reliability analysis
WI-PSO	1. Achieve global optima 2. Significant exploration 3.Significant exploitation 4. Low mean loss 5. Significant for multi modal functions 6. Superior performance on real world applications	1. Poor swarm diversity

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3002725, IEEE Access

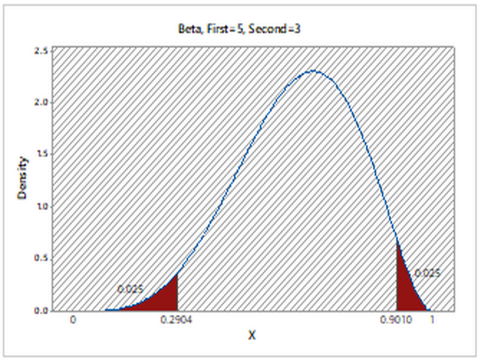

FIGURE 1: Density Plot for Exponential distribution

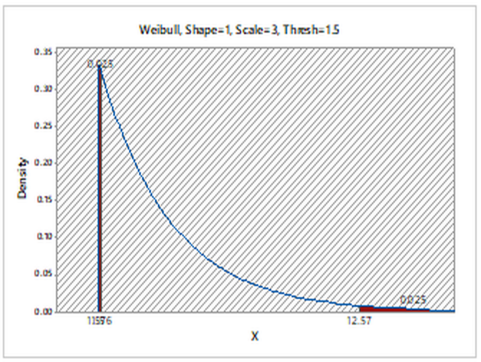

FIGURE 5: Density Plot for Weibull distribution

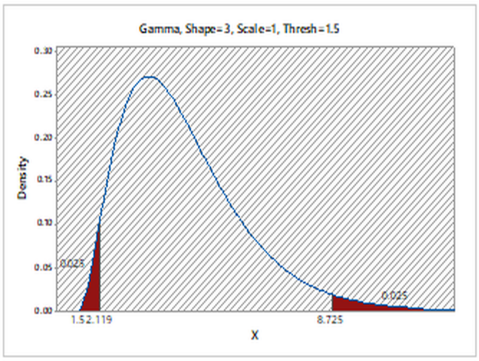

FIGURE 2: Density Plot for Beta distribution

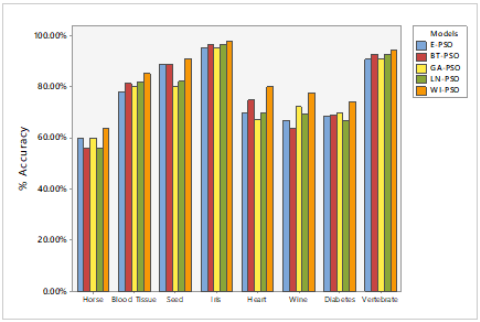

FIGURE 6: T
esting Accuracy on eight bench mark data sets

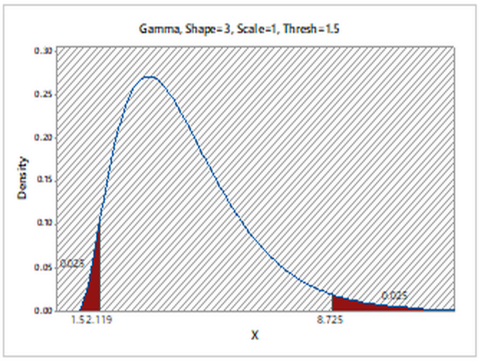

FIGURE 3: Density Plot for Gamma distribution

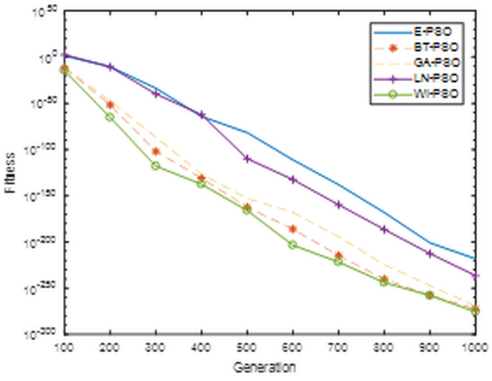

FIGURE 7: Fitness curve for f1

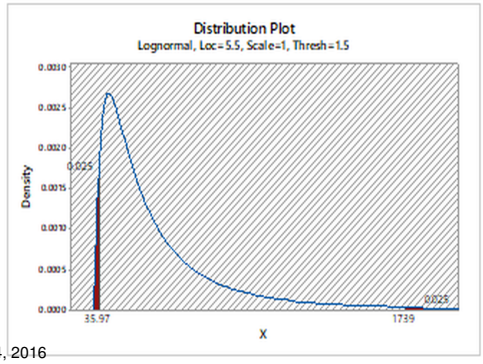

FIGURE 4: Density Plot for Lognormal distribution.

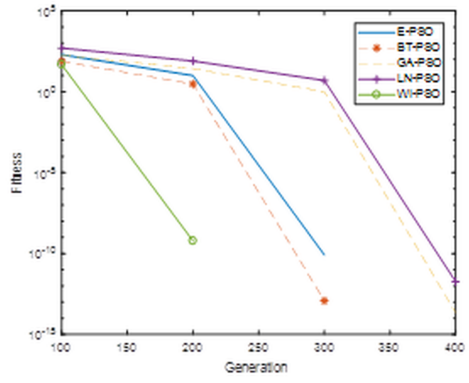

FIGURE 8: Fitness curve for f2

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3002725, IEEE Access

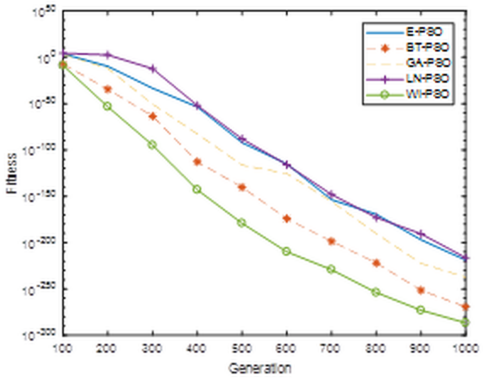

FIGURE 9: Fitness curve for f3

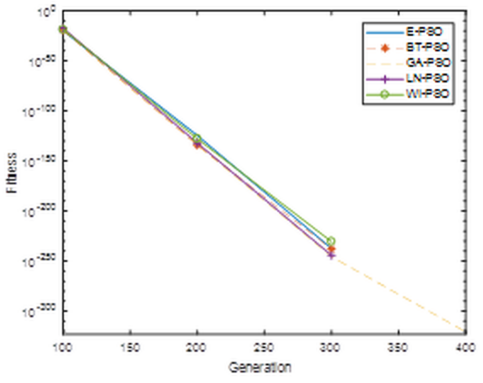

FIGURE 10: Fitness curve for f4

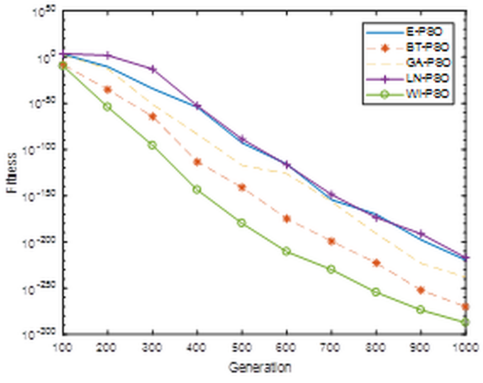

FIGURE 11: Fitness curve for f5

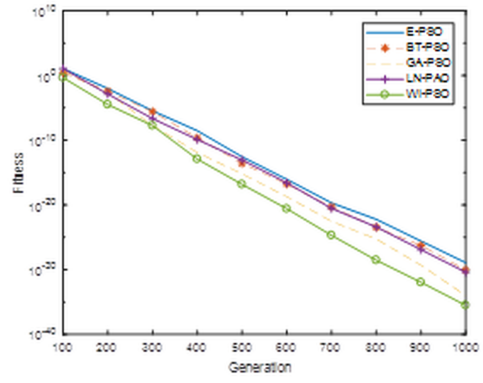

FIGURE 12: Fitness curve for f8

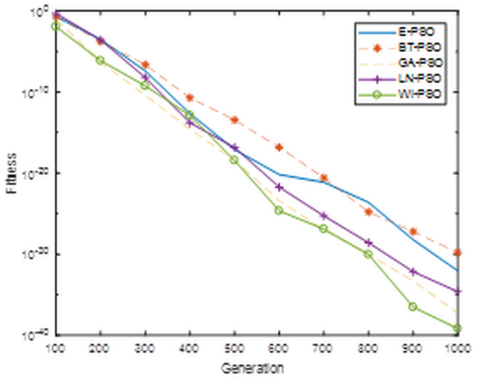

FIGURE 13: Fitness curve for f9

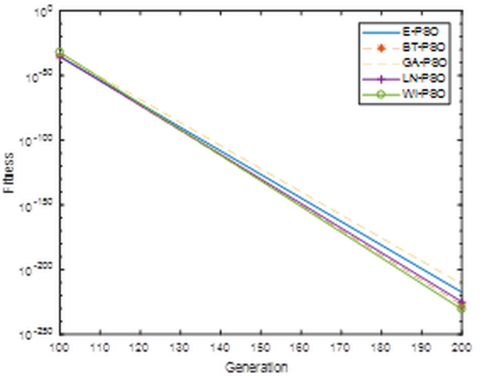

FIGURE 14: Fitness curve for f10

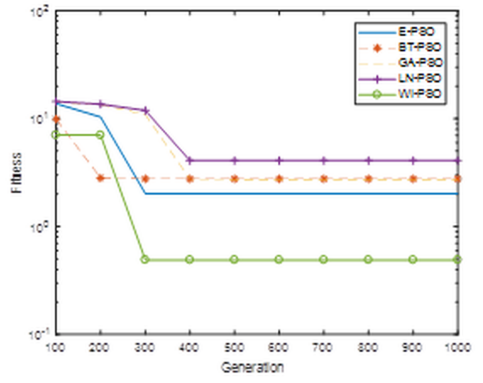

FIGURE 15: Fitness curve for f11

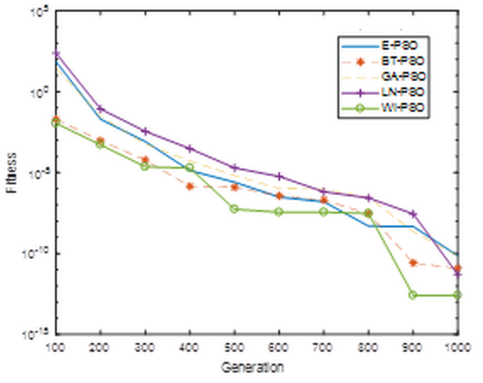

FIGURE 16: Fitness curve for f16

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3002725, IEEE Access

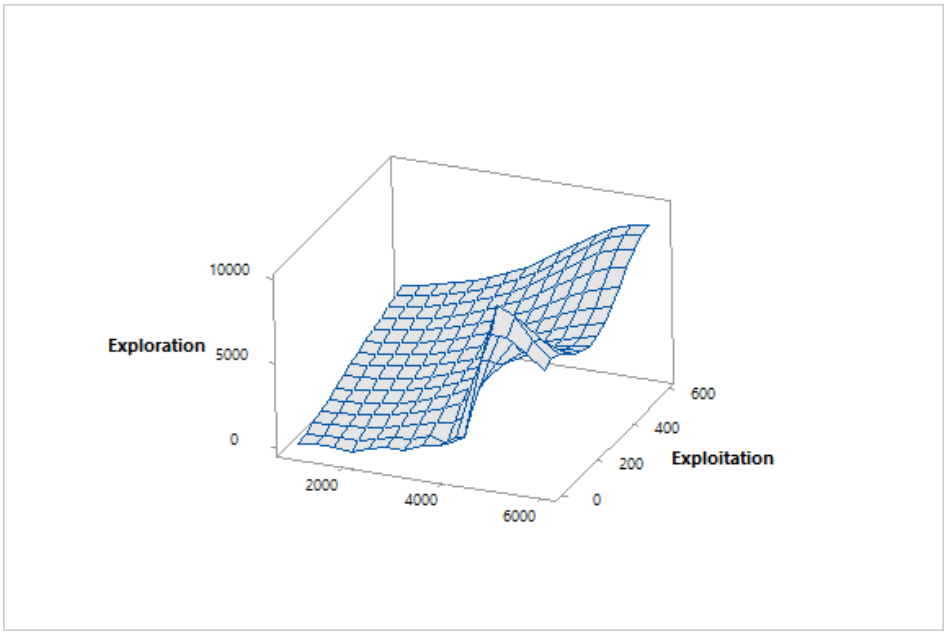

FIGURE 17: Surface Plot comparison of weibull to PSO in terms of Exploration vs Exploitation

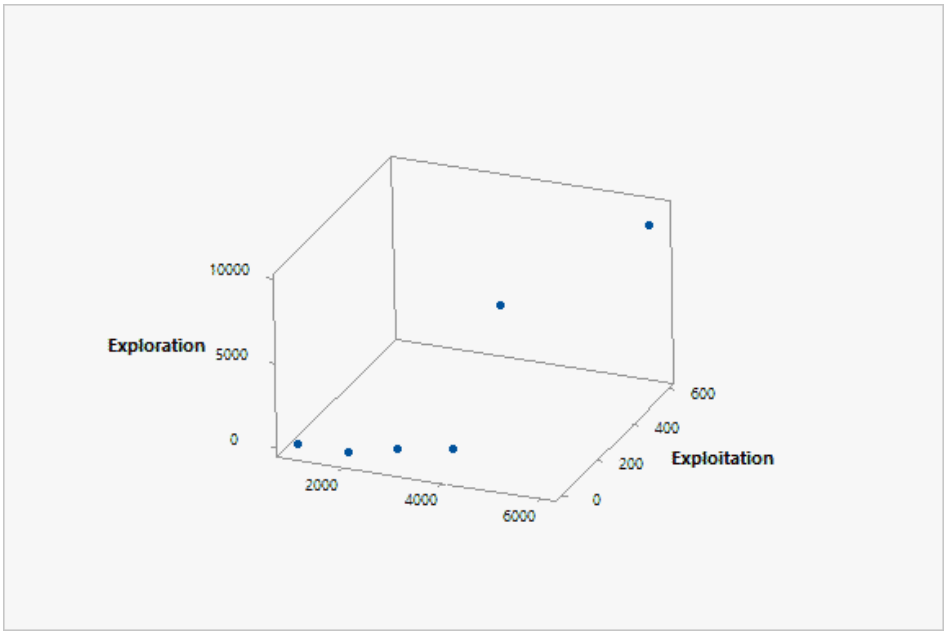

FIGURE 18: Scatter Plot comparison of weibull to PSO in terms of Exploration vs Exploitation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3002725, IEEE Access

TABLE 2: List of Standard Benchmark Functions

F#	Function Name	Objective Function	Search space(S)	(f_{min})
F1	**Sphere**	$Min\ f(x) = \sum_{i=1}^{n} x_i^2$	$-5.12 \leq x_i \leq 5.12$	0
F2	**Rastrigin**	$Min\ f(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos?(2\pi x) + 10]_i$	$-5.12 \leq x_i \leq 5.12$	0
F3	**Moved axis parallel hyper-ellipsoid**	$Min\ f(x) = \sum_{i=1}^{n} 5i.x_i^2$	$-5.12 \leq x_i \leq 5.12$	0
F4	**Rotated hyper-ellipsoid**	$Min\ f(x) = \sum_{i}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	$-65.536 \leq x_i \leq 65.536$	0
F5	**Axis parallel hyper-ellipsoid**	$Min\ f(x) = \sum_{i=1}^{n} i.x_i^2$	$-5.12 \leq x_i \leq 5.12$	0
F6	**Schwefel**	$Min\ f(x) = \sum_{i=1}^{n} x_i^{\alpha}$	$-100 \leq x_i \leq 100$	0
F7	**Schwefel 1.2**	$Min\ f(x) = \sum_{i}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$	$-100 \leq x_i \leq 100$	0
F8	**Schwefel** 2.21	$Minf(x) = \left(\max_{1<i<D} \lvert x_i \rvert\right)$	$-100 \leq x_i \leq 100$	0
F9	**Schwefel** 2.22	$Min\ f(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$-100 \leq x_i \leq 100$	0
F10	**Schwefel** 2.23	$Min\ f(x) = \sum_{i=1}^{n} x_i^{10}$	$-10 \leq x_i \leq 10$	0
F11	**Schaffer**	$Min\ f(x) = \sum_{i=1}^{n} 0.5 + \frac{sin^2\sqrt{x_i^2+x_{i+1}^2}-0.5}{\left[1+0.001\left(x_i^2+x_{i+1}^2\right)\right]^2}$	$-100 \leq x_i \leq 100$	0
F12	**Chung Reynolds**	$Min\ f(x) = \left(\sum_{i=1}^{n} x_i^2\right)^2$	$-100 \leq x_i \leq 100$	0
F13	**Sum of different power**	$Min\ f(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	$-1 \leq x_i \leq 1$	0
F14	**Csendes**	$Min\ f(x) = \sum_{i=1}^{n} x_i^6 (2 + sin\frac{1}{x_i})$	$-1 \leq x_i \leq 1$	0
F15	**Schumer Steiglitz**	$Min\ f(x) = \sum_{i=1}^{n} x_i^4$	$-5.12 \leq x_i \leq 5.12$	0
F16	**Dejong's function with noise**	$Min\ f(x) = \left(\sum_{i=0}^{n-1} (i+1)x_i^4\right) + rand[0,1]$	$-1.28 \leq x_i \leq 1.28$	0

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3002725, IEEE Access

TABLE 3: Comparative results of proposed algorithm WI-PSO with other existing algorithms. Mean indicates the mean fitness and Std. Dev represents standard deviation

F#	DIM	E-PSO		BT-PSO		GA-PSO		LN-PSO		WI-PSO	
		Mean	**Std. Dev**	**Mean**	**Std. Dev**	**Mean**	**Std. Dev**	**Mean**	**Std. Dev**	**Mean**	**Std. Dev**
F1	**10**	2.05E-76	6.15E-76	2.04E-78	6.13E-78	1.39E-77	4.17E-77	7.86E+00	2.36E+01	**2.03E-78**	**6.10E-78**
	20	2.62E+00	7.86E+00	3.12E-88	9.35E-88	9.35E-88	6.44E-87	1.05E+01	3.15E+01	**7.10E-89**	**2.13E-88**
	30	2.62E+00	7.86E+00	6.78E-49	2.03E-48	4.96E-45	1.49E-44	2.88E+01	8.65E+01	**3.32E-66**	**9.95E-66**
F2	**10**	1.89E+00	5.67E+00	7.96E-01	2.39E+00	1.19E+00	3.58E+00	1.34E+01	3.81E+01	**4.97E-01**	**1.49E+00**
	20	8.16E+00	2.32E+01	1.69E+00	5.07E+00	1.20E+01	3.42E+01	3.93E+01	1.08E+02	**1.19E+00**	**3.58E+00**
	30	2.05E+01	5.83E+01	7.86E+00	2.26E+01	2.11E+01	5.46E+01	5.82E+01	1.49E+02	**4.78E+00**	**1.43E+01**
F3	**10**	3.36E-82	1.01E-81	**5.41E-83**	**1.62E-82**	2.10E-77	6.30E-77	1.31E+02	3.93E+02	7.25E-82	2.17E-81
	20	9.18E+01	2.75E+02	1.59E-91	4.77E-91	1.18E+02	3.54E+02	3.54E+02	2.36E+03	**3.73E-96**	**1.12E-95**
	30	4.85E+02	1.45E+03	1.97E-30	5.91E-30	1.44E+02	4.33E+02	2.27E+03	6.72E+03	**1.40E-52**	**4.21E-52**
F4	**10**	5.66E-146	1.70E-145	9.39E-150	9.39E-150	8.93E-143	2.68E-142	4.73E-144	1.42E-143	**2.79E-151**	**8.38E-151**
	20	6.82E-153	2.04E-152	6.80E-155	2.04E-154	3.03E-151	9.08E-151	6.12E-148	1.84E-147	**2.03E-156**	**6.09E-156**
	30	1.66E-125	4.97E-125	4.46E-135	1.34E-134	9.43E-135	2.83E-134	5.50E-133	1.65E-132	**3.08E-135**	**9.23E-135**
F5	**10**	6.73E-83	2.02E-82	**1.08E-83**	**3.24E-83**	4.20E-78	1.26E-77	2.62E+01	7.86E+01	1.45E-82	4.35E-82
	20	1.84E+01	5.51E+01	3.18E-92	9.54E-92	2.36E+01	7.08E+01	7.08E+01	4.72E+02	**7.45E-97**	**2.24E-96**
	30	9.70E+01	2.91E+02	3.94E-31	1.18E-30	2.88E+01	8.65E+01	4.54E+02	1343.14	**2.81E-53**	**8.42E-53**
F6	**10**	3.25E-193	0.00E+00	2.45E-195	0.00E+00	2.45E-195	0.00E+00	1.92E-192	0.00E+00	**1.49E-197**	0.00E+00
	20	**4.15E-195**	0.00E+00	9.33E-186	0.00E+00	1.62E-188	0.00E+00	1.58E-187	0.00E+00	3.25E-193	0.00E+00
	30	9.59E-158	2.88E-157	1.43E-160	4.30E-160	3.84E-163	0.00E+00	2.84E-163	0.00E+00	**2.02E-169**	0.00E+00
F7	**10**	8.43E-04	2.53E-03	3.51E-02	1.05E-01	8.62E-04	2.59E-03	1.35E-03	4.04E-03	**4.65E-04**	**1.40E-03**
	20	5.73E+00	1.72E+01	2.28E+00	6.83E+00	4.33E+00	1.30E+01	3.88E+00	1.16E+01	**1.43E+00**	**6.28E+00**
	30	1.94E+01	5.83E+01	1.26E+01	3.78E+01	3.25E+01	9.72E+01	8.10E+00	2.43E+01	**6.97E+00**	**2.09E+01**
F8	**10**	9.41E-28	2.82E-27	5.57E-27	1.67E-26	1.47E-26	4.40E-26	2.28E-25	6.85E-25	**5.19E-29**	**1.56E-28**
	20	9.35E-08	2.81E-07	4.57E-10	1.37E-09	5.37E-09	1.61E-08	2.97E-08	8.90E-08	**8.36E-12**	**2.51E-11**
	30	8.94E-02	2.68E-01	3.58E-02	1.07E-01	2.13E-02	6.40E-02	8.13E-03	2.44E-02	**9.55E-04**	**2.86E-03**
F9	**10**	1.71E-38	5.12E-38	2.88E-38	8.64E-38	5.50E-39	1.65E-38	2.09E-38	6.26E-38	**1.18E-39**	**3.54E-39**
	20	1.99E-07	5.96E-07	1.73E-07	5.19E-07	6.77E-08	2.03E-07	1.80E-12	1.80E-12	**1.78E-14**	**5.33E-14**
	30	2.13E-02	6.38E-02	4.45E-03	1.33E-02	5.30E-03	1.59E-02	4.87E-02	1.46E-01	**1.11E-04**	**3.31E-04**
F10	**10**	2.11E-313	0.00E+00	2.31E-300	0.00E+00	6.90E-312	0.00E+00	2.54E-305	0.00E+00	**1.82E-313**	**2.11E-313**
	20	3.84E-263	0.00E+00	6.65E-266	0.00E+00	7.78E-257	0.00E+00	1.56E-259	0.00E+00	**1.42E-275**	**3.84E-263**
	30	1.20E-205	0.00E+00	4.13E-216	0.00E+00	2.14E-204	0.00E+00	1.17E-210	0.00E+00	**5.60E-219**	**1.20E-205**
F11	**10**	1.12E+00	1.67E+00	4.42E-01	8.31E-01	1.02E+00	1.48E+00	1.47E+00	1.86E+00	**7.47E-02**	**1.51E-01**
	20	2.44E+00	3.18E+00	2.90E+00	1.02E+00	2.14E+00	3.43E+00	4.06E+00	2.36E+00	**5.11E-01**	**9.36E-01**
	30	4.03E+00	4.09283	3.49E+00	2.15E+00	5.79E+00	4.72E+00	6.86E+00	4.27E+00	**1.61E+00**	**2.09E+00**
F12	**10**	2.25E-151	6.75E-151	2.39E-160	7.18E-160	3.36E-160	1.01E-159	4.07E-154	1.22E-153	**1.18E-160**	**3.53E-160**
	20	1.21E-179	0.00E+00	9.68E-183	0.00E+00	5.66E-177	0.00E+00	8.20E-181	0.00E+00	**6.71E-185**	**0.00E+00**
	30	2.80E-87	8.40E-87	1.02E-111	3.05E-111	6.71E-59	2.01E-58	1.23E-102	3.68E-102	**2.73E-134**	**8.18E-134**
F13	**10**	**6.67E-67**	**2.00E-66**	6.87E-67	2.06E-66	3.93E-65	1.18E-64	8.63E-64	2.59E-63	1.13E-66	3.39E-66
	20	1.00E-01	3.00E-01	2.19E-109	6.58E-109	1.20E-105	3.61E-105	2.00E-01	6.00E-01	**4.86E-114**	**1.46E-113**
	30	1.84E-133	5.51E-133	5.12E-121	1.53E-120	2.34E-131	7.03E-131	2.00E-01	6.00E-01	**3.36E-138**	**1.01E-137**
F14	**10**	3.14E-194	0.00E+00	1.50E-194	0.00E+00	1.04E-189	0.00E+00	1.29E-194	0.00E+00	**2.98E-202**	0.00E+00
	20	5.38E-143	1.62E-142	7.11E-94	2.13E-93	6.09E-14	1.83E-13	1.16E-01	3.48E-01	**7.64E-149**	**2.29E-148**
	30	3.54E-09	1.06E-08	1.14E-08	3.43E-08	1.78E-09	5.34E-09	6.95E-01	2.09E+00	**8.24E-11**	**2.47E-10**
F15	**10**	2.44E-143	7.31E-143	7.12E-137	2.14E-136	7.51E-139	2.25E-138	1.37E+02	4.12E+02	**3.37E-145**	**1.01E-144**
	20	2.22E-148	6.65E-148	6.07E-147	1.82E-146	9.31E-152	2.79E-151	1.37E+02	4.12E+02	**3.38E-152**	**1.01E-151**
	30	5.29E-123	1.59E-122	7.26E-125	2.18E-124	6.87E+01	2.06E+02	2.75E+02	8.25E+02	**1.11E-132**	**3.33E-132**
F16	**10**	9.67E-05	2.32E-04	1.51E-04	3.86E-04	6.13E-05	1.09E-04	1.34E+00	4.03E+00	**3.31E-05**	**1.52E-04**
	20	1.55E-03	4.38E-03	3.14E-04	9.18E-04	4.19E-04	1.12E-03	8.86E+00	2.66E+01	**1.99E-04**	**5.72E-04**
	30	7.25E+00	2.17E+01	1.26E-03	3.54E-03	2.96E+00	8.86E+00	2.42E+01	7.25E+01	**8.38E-04**	**2.38E-03**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3002725, IEEE Access

TABLE 4: Description of dataset

S. No	Data Set	Features		Nature	No. of Inputs	No. of Classes	Sample Per Class	Total Samples
		Continuous	Disc					
1	Horse	27	-	Real	27	2	$C_1 = 12, C_2 = 24$	36
2	Blood Tissue	5	-	Real	5	2	$C_1 = 56, C_2 = 18$	74
3	Seed	7	-	Real	7	3	$C_1 = 14, C_2 = 14, C_3 = 14$	42
4	Iris	4	-	Real	4	3	$C_1 = 50, C_2 = 50, C_3 = 50$	150
5	Heart	13	-	Real	13	2	$C_1 = 12, C_2 = 15$	27
6	Wine	13	-	Real	13	3	$C_1 = 8, C_2 = 13, C_3 = 7$	28
7	Diabetes	8	-	Real	8	2	$C_1 = 25, C_2 = 51$	76
8	Vertebral	6	-	Real	6	2	$C_1 = 28, C_2 = 14$	42

TABLE 5: Accuracy results for 10-fold classification rates of 8 datasets.

DataSets	BPA		E-PSO		BT-PSO		GA-PSO		LN-PSO		WI-PSO	
	Tr. Acc	Ts. Acc	Tr. Acc	Ts. Acc	Tr. Acc	Ts. Acc	Tr. Acc	Ts. Acc	Tr. Acc	Ts. Acc	Tr. Acc	Ts. Acc
Horse	64.4%	57.8%	75.8%	60%	71.2%	56%	70.9%	60%	71.6%	56%	75.1%	**64%**
Blood Tissue	76.3%	73.4%	90.6%	78%	93.3%	81.3%	91.1%	80%	88.9%	82%	92.8%	**85.3%**
Seed	84.2%	77.5%	96.3%	88.8%	98.1%	88.8%	98.1%	80%	98.7%	82.2%	98.1%	**91.1%**
Iris	98.2%	95.6%	95.5%	95.3%	98.8%	96.6%	98.8%	95.3%	98.6%	96.6%	99.1%	**98%**
Heart	78.5%	68.3%	94.7%	70%	99.5%	75%	99.1%	67.5%	99.1%	70%	98.6%	**80%**
Wine	67.3%	62.0%	78.7%	66.6%	74.2%	63.8%	79.5%	72.2%	84.0%	69.4%	84.0%	**77.7%**
Diabetes	86.1%	65.3%	94.2%	68.3%	96.0%	69.1%	92.3%	70%	90.2%	66.6%	91.4%	**74.1%**
Vertebrl	91.4%	84.9%	92.0%	91.0%	98.0%	92.8%	98.0%	91.0%	98.8%	92.8%	98.0%	**94.6%**

REFERENCES

[1] J. Liu, H. A. Abbass, and K. C. Tan, "Evolutionary computation," in Evolutionary Computation and Complex Networks. Springer, 2019, pp. 3–22.

[2] X. Liang, W. Li, Y. Zhang, and M. Zhou, "An adaptive particle swarm optimization method based on clustering," Soft Computing, vol. 19, no. 2, pp. 431–448, 2015.

[3] X.-X. Ma and J.-S. Wang, "Optimized parameter settings of binary bat algorithm for solving function optimization problems," Journal of Electrical and Computer Engineering, vol. 2018, 2018.

[4] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," Mechanical Systems and Signal Processing, vol. 115, pp. 213–237, 2019.

[5] T. R. Patil, S. Sherekar et al., "Performance analysis of naive bayes and j48 classification algorithm for data classification," International journal of computer science and applications, vol. 6, no. 2, pp. 256–261, 2013.

[6] R. Tanno, K. Arulkumaran, D. C. Alexander, A. Criminisi, and A. Nori, "Adaptive neural trees," arXiv preprint arXiv:1807.06699, 2018.

[7] D. Pebrianti, N. Q. Ann, L. Bayuaji, N. H. Abdullah, Z. M. Zain, and I. Riyanto, "Extended bat algorithm (eba) as an improved searching optimization algorithm," in Proceedings of the 10th National Technical Seminar on Underwater System Technology 2018. Springer, 2019, pp. 229–237.

[8] E. I. Ojanguren, "Computational intelligence," in Neuro-fuzzy Modeling of Multi-field Surface Neuroprostheses for Hand Grasping. Springer, 2019, pp. 69–87.

[9] X. Li and M. Clerc, "Swarm intelligence," in Handbook of Metaheuristics. Springer, 2019, pp. 353–384.

[10] J. Kennedy, "Swarm intelligence," in Handbook of nature-inspired and innovative computing. Springer, 2006, pp. 187–219.

[11] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, and Q. Pan, "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 4, pp. 904–916, 2019.

[12] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in Robots and biological systems: towards a new bionics? Springer, 1993, pp. 703–712.

[13] A. Engelbrecht, "Fundamentals of computational swarm intelligence," Hoboken: John Wiley & Sons, Ltd, 2006.

[14] V. Bahel, A. Peshkar, and S. Singh, "Swarm intelligence-based systems: A review," in Proceeding of International Conference on Computational Science and Applications. Springer, 2020, pp. 149–156.

[15] J. R. Manne, "Swarm intelligence for multi-objective optimization in engineering design," in Advanced Methodologies and Technologies in Artificial Intelligence, Computer Simulation, and Human-Computer Interaction. IGI Global, 2019, pp. 180–194.

[16] R. Eberhart and J. Kennedy, "Particle swarm optimization," in Proceedings of the IEEE international conference on neural networks, vol. 4. Citeseer, 1995, pp. 1942–1948.

[17] X. Gong, X. Zheng, J. Fang, and G. Liu, "Optimized layout methods based on optimization algorithms for dpos," Aerospace Science and Technology, vol. 84, pp. 484–496, 2019.

[18] C. Census, H. Wang, J. Zhang, P. Deng, and T. Li, "Particle subswarms collaborative clustering," IEEE Transactions on Computational Social Systems, vol. 6, no. 6, pp. 1165–1179, 2019.

[19] S. Walczak, "Artificial neural networks," in Advanced Methodologies and Technologies in Artificial Intelligence, Computer Simulation, and Human-Computer Interaction. IGI Global, 2019, pp. 40–53.

[20] H. Yuan, J. Bi, and M. Zhou, "Spatiotemporal task scheduling for heterogeneous delay-tolerant applications in distributed green data centers," IEEE Transactions on Automation Science and Engineering, vol. 16, no. 4, pp. 1686–1697, 2019.

[21] G. Ghosh, P. Mandal, and S. C. Mondal, "Modeling and optimization of surface roughness in keyway milling using ann, genetic algorithm, and particle swarm optimization," The International Journal of Advanced Manufacturing Technology, vol. 100, no. 5-8, pp. 1223–1242, 2019.

[22] J. Wang and T. Kumbasar, "Parameter optimization of interval type-2 fuzzy neural networks based on pso and bbbc methods," IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 1, pp. 247–257, 2019.

[23] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," IEEE transactions on neural networks and learning systems, vol. 30, no. 2, pp. 601–614, 2018.

[24] W. F. Sacco and A. C. Rios-Coelho, "On initial populations of differential evolution for practical optimization problems," in Computational Intelligence, Optimization and Inverse Problems with Applications in Engineering. Springer, 2019, pp. 53–62.

[25] T. Maini, A. Kumar, R. K. Misra, and D. Singh, "Fuzzy rough set-based feature selection with improved seed population in pso and ids," in Computational Intelligence: Theories, Applications and Future Directions-Volume II. Springer, 2019, pp. 137–149.

[26] R. V. Hogg and A. T. Craig, "Introduction to mathematical statistics.(5th edition)," Englewood Hills, New Jersey, 1995.

[27] R. Thangaraj, M. Pant, and K. Deep, "Initializing pso with probability distributions and low-discrepancy sequences: the comparative results," in 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). IEEE, 2009, pp. 1121–1126.

[28] D. KNUTH, "The art of computer programming 1: Fundamental algorithms 2: Seminumerical algorithms 3: Sorting and searching," Cité en, p. 168, 1968.

[29] C. J. Merz and P. M. Murphy, "Uci repository of machine learning databases, 1998," 1996.

[30] N. Q. Uy, N. X. Hoai, R. I. McKay, and P. M. Tuan, "Initialising pso with randomised low-discrepancy sequences: the comparative results," in 2007 IEEE Congress on Evolutionary Computation. IEEE, 2007, pp. 1985–1992.

[31] J. Zhang, X. Zhu, Y. Wang, and M. Zhou, "Dual-environmental particle swarm optimizer in noisy and noise-free environments," IEEE transactions on cybernetics, vol. 49, no. 6, pp. 2011–2021, 2018.

[32] R. Brits, A. P. Engelbrecht, and F. Van den Bergh, "A niching particle swarm optimizer," in Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning, vol. 2. Singapore: Orchid Country Club, 2002, pp. 692–696.

[33] J. VAN DER COPUT, "Verteilungsfunktionen i & ii," in Nederl. Akad. Wetensch. Proc., vol. 38, 1935, pp. 1058–1066.

[34] R. A. Krohling and L. dos Santos Coelho, "Pso-e: Particle swarm with exponential distribution," in 2006 IEEE International Conference on Evolutionary Computation. IEEE, 2006, pp. 1428–1433.

[35] J. Li, J. Zhang, C. Jiang, and M. Zhou, "Composite particle swarm optimizer with historical memory for function optimization," IEEE transactions on cybernetics, vol. 45, no. 10, pp. 2350–2363, 2015.

[36] M. Pant, R. Thangaraj, and A. Abraham, "Particle swarm optimization using adaptive mutation," in 2008 19th International Workshop on Database and Expert Systems Applications. IEEE, 2008, pp. 519–523.

[37] Y. Cao, H. Zhang, W. Li, M. Zhou, Y. Zhang, and W. A. Chaovalitwongse, "Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions," IEEE Transactions on Evolutionary Computation, vol. 23, no. 4, pp. 718–731, 2018.

[38] K. Parsopoulos and M. Vrahatis, "Initializing the particle swarm optimizer using the nonlinear simplex method," Advances in intelligent systems, fuzzy systems, evolutionary computation, vol. 216, pp. 1–6, 2002.

[39] Q. Kang, C. Xiong, M. Zhou, and L. Meng, "Opposition-based hybrid strategy for particle swarm optimization in noisy environments," IEEE Access, vol. 6, pp. 21 888–21 900, 2018.

[40] M. Richards and D. Ventura, "Choosing a starting configuration for particle swarm optimization," in IEEE Int. Joint. Conf. Neural, vol. 3, 2004, pp. 2309–2312.

[41] P. Roy, G. S. Mahapatra, and K. N. Dey, "Forecasting of software reliability using neighborhood fuzzy particle swarm optimization based novel neural network," IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 6, pp. 1365–1383, 2019.

[42] H. Jabeen, Z. Jalil, and A. R. Baig, "Opposition based initialization in particle swarm optimization (o-pso)," in Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers. ACM, 2009, pp. 2047–2052.

[43] Z. Lv, L. Wang, Z. Han, J. Zhao, and W. Wang, "Surrogate-assisted particle swarm optimization algorithm with pareto active learning for expensive multi-objective optimization," IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 3, pp. 838–849, 2019.

[44] A. Gutiérrez, M. Lanza, I. Barriuso, L. Valle, M. Domingo, J. Perez, and J. Basterrechea, "Comparison of different pso initialization techniques for high dimensional search space problems: A test with fss and antenna arrays," in Proceedings of the 5th European Conference on Antennas and Propagation (EUCAP). IEEE, 2011, pp. 965–969.

[45] I. De Falco, A. Della Cioppa, and E. Tarantino, "Facing classification problems with particle swarm optimization," Applied Soft Computing, vol. 7, no. 3, pp. 652–658, 2007.

[46] A. Subasi, "Classification of emg signals using pso optimized svm for diagnosis of neuromuscular disorders," Computers in biology and medicine, vol. 43, no. 5, pp. 576–586, 2013.
[47] Z. Liu, P. Zhu, W. Chen, and R.-J. Yang, "Improved particle swarm optimization algorithm using design of experiment and data mining techniques," Structural and Multidisciplinary Optimization, vol. 52, no. 4, pp. 813–826, 2015.
[48] X. Xu, J. Li, M. Zhou, J. Xu, and J. Cao, "Accelerated two-stage particle swarm optimization for clustering not-well-separated data," IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018.
[49] G. P. Zhang, "Neural networks for classification: a survey," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 30, no. 4, pp. 451–462, 2000.
[50] S. Haykin, Neural networks. Prentice hall New York, 1994, vol. 2.
[51] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy et al., "Evolving deep neural networks," in Artificial Intelligence in the Age of Neural Networks and Brain Computing. Elsevier, 2019, pp. 293–312.
[52] M. Castellani, "Evolutionary generation of neural network classification empirical comparison," Neurocomputing, vol. 99, pp. 214–229, 2013.
[53] J. Yang and J. Ma, "Feed-forward neural network training using sparse representation," Expert Systems with Applications, vol. 116, pp. 255–264, 2019.
[54] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.
[55] A. W. Marshall and I. Olkin, "A multivariate exponential distribution," Journal of the American Statistical Association, vol. 62, no. 317, pp. 30–44, 1967.
[56] A. K. Gupta and S. Nadarajah, Handbook of beta distribution and its applications. CRC press, 2004.
[57] E. W. Stacy et al., "A generalization of the gamma distribution," The Annals of mathematical statistics, vol. 33, no. 3, pp. 1187–1192, 1962.
[58] L. Klein, "The lognormal distribution." 1957.
[59] M. Fréchet, "Sur la loi de probabilité de l'écart maximum," Ann. Soc. Math. Polon., vol. 6, pp. 93–116, 1927.
[60] P. Rosin, "Laws governing the fineness of powdered coal," Journal of Institute of Fuel, vol. 7, pp. 29–36, 1933.