Contents lists available at ScienceDirect

Expert Systems With Applications

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

Application of mutation operators to flower pollination algorithm

Rohit Salgotra*, Urvinder Singh

Department of ECE, Thapar University, Punjab, India

ARTICLE INFO

Article history:
Received 1 December 2016
Revised 23 February 2017
Accepted 24 February 2017
Available online 27 February 2017

Keywords:
Gaussian mutation
Cauchy mutation
Adaptive Levy mutation
Mean mutation
Adaptive mean mutation
Combined mutation
Flower pollination algorithm

ABSTRACT

Flower pollination algorithm (FPA) is a recent addition to the field of nature inspired computing. The algorithm has been inspired from the pollination process in flowers and has been applied to a large spectra of optimization problems. But it has certain drawbacks which prevents its applications as a standard algorithm. This paper proposes new variants of FPA employing new mutation operators, dynamic switching and improved local search. A comprehensive comparison of proposed algorithms has been done for different population sizes for optimizing seventeen benchmark problems. The best variant among these is adaptive-Lévy flower pollination algorithm (ALFPA) which has been further compared with the well-known algorithms like artificial bee colony (ABC), differential evolution (DE), firefly algorithm (FA), bat algorithm (BA) and grey wolf optimizer (GWO). Numerical results show that ALFPA gives superior performance for standard benchmark functions. The algorithm has also been subjected to statistical tests and again the performance is better than the other algorithms.

1. Introduction

Nature has always served as an inspiration for solving our problems since evolution which can be observed from various natural phenomenon like decision making, learning, perception, immune systems and others. As these phenomena depict nature inspired computing, various computational methods known as meta−heuristic optimization algorithms have been derived. These algorithms are in use for the past two decades and are becoming more and more popular in almost every field of research. This is due to the fact that computationally intelligent algorithms are faster as well as flexible when compared to classical computing techniques and are applicable even in case of failure of classical techniques (Gutjahr, 2009).

Nature inspired computing uses the basis of biological systems or natural phenomenon to solve optimization problems. A large number of algorithms have been proposed in this context. These algorithms are mainly grouped into two categories: swarm intelligent algorithms and evolutionary algorithms. Swarm intelligent algorithms are based on the social behaviour of group of animals. The algorithms in this group include particle swarm optimization (PSO) (Eberhart & Kennedy, 1995) (inspired from the bird flocking behaviour), ant colony optimization (ACO) (Dorigo, Birattari, & Stutzle, 2006) (inspired from the social intelligence of colonial ants to find the closest path between food source and nest), artificial bee colony (ABC) (Karaboga & Basturk, 2007) (based on the swarming behaviour of honey bees), cuckoo search (CS) (Yang, 2010a), bat algorithm (BA) (Yang, 2010a), firefly algorithm (FA) (Yang, 2010b) and grey wolf optimizer (Mirjalili, Mirjalili, & Lewis, 2014). Evolutionary algorithms on the other hand are based on the laws of evolution. The basis for these algorithms is that best individuals from the parent generation are combined together to form the child or next generation. The algorithms in this group include genetic algorithm (GA) (Holland, 1992), differential evolution (DE) (Storn, & Price, 1995), genetic programming (Koza, 1992), biogeography based optimization (BBO) (Simon, 2008), evolutionary strategy (ES) (Rechenberg, 1978), ant lion algorithm (ALO) (Mirjalili, 2015), moth search algorithm (Gang, 2016) and others.

Flower pollination algorithm (FPA) (Yang, 2012) is one such algorithm introduced in the recent past. The algorithm is based on the pollination process in plants and has been explicitly discussed in Section 2. It has been applied to various applications in different fields of research (Sharawi, Emary, Saroit, & El-Mahdy, 2014; Prathiba, Moses, & Sakthivel, 2014; Singh & Salgotra, 2016). The algorithm though is very promising but has not delivered up to the expectations and it has been shown qualitatively as well as quantitatively by Draa (2015). An algorithm in its basic version is not state-of-the art and a lot of modifications are applied to it to make it one. In case of FPA, five new algorithms based upon FPA were proposed in Draa (2015). The proposed versions were opposition-based FPA (OFPA), generalised opposition-based FPA (GOFPA), modified FPA (MFPA) using enhanced global pollination equation, modified opposition based FPA (MOFPA) and modified generalised opposition-based FPA (MGOFPA). The last two vari-

* Corresponding author.
E-mail addresses: r.03dec@gmail.com (R. Salgotra), urvinders@gmail.com (U. Singh).

http://dx.doi.org/10.1016/j.eswa.2017.02.035
0957-4174/© 2017 Elsevier Ltd. All rights reserved.

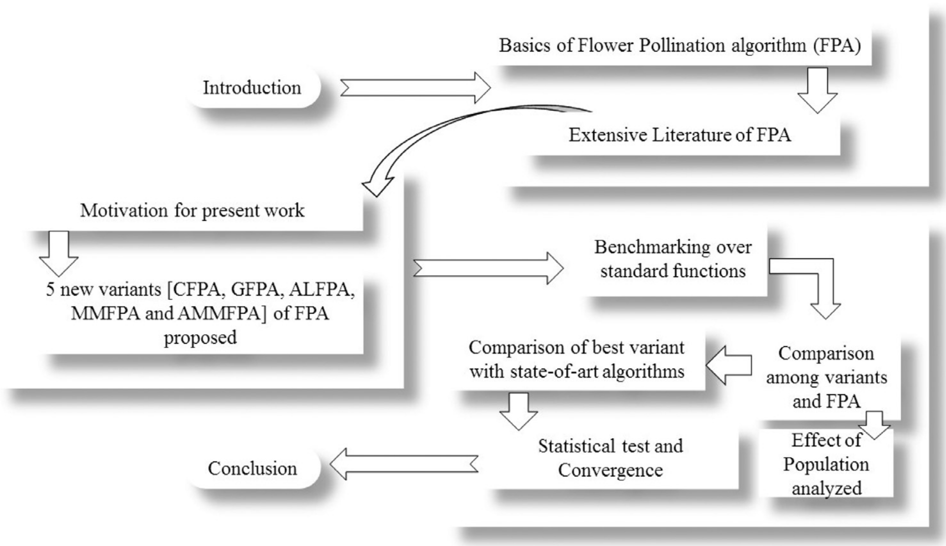
Basics of Flower Pollination algorithm (FPA)
Introduction
Extensive Literature of FPA
Motivation for present work
5 new variants [CFPA, GFPA, ALFPA, MMFPA and AMMFPA] of FPA proposed
Benchmarking over standard functions
Comparison of best variant with state-of-art algorithms
Comparison among variants and FPA
Effect of Population analyzed
Statistical test and Convergence
Conclusion

Fig. 1. Outline of the article.

ants were proposed by equation modification of the first two variants. The authors further analysed their proposed versions with respect to the standard algorithm and found that these variants improve the performance of FPA but still have some limitations and we can't term any among them as standard algorithm.

From the literature discussed in Section 3, it can be said that FPA is still to be explored and so in present work, detailed effect of mutation operators has been discussed and based upon mutation operators, new variants of FPA are proposed in Section 4. These variants use mutation operators like Gaussian, Cauchy, mean mutation, adaptive mean mutation and adaptive Lévy to add a modification to the basic FPA. Another modification is in the switch probability being used dynamically and justification for its use is given in Section 3. Further the FPA variants are also compared with respect to the mutation operators of DE and an explicit discussion about the similarity between the operators of DE and FPA is analysed. The same has been discussed in Section 4. The variants are analysed in terms of different population size, dynamic switch probability, statistical testing and are applied to standard benchmark problems. A detailed comparative study with respect to other state of art is also discussed. The result and discussion part is in Section 5. The concluding section is Section 6 and it provides a brief review of the analysis of results. The detailed outline of the paper is given in Fig 1.

2. Flower pollination algorithm

Flowers are the most fascinating creatures on earth and have been found on earth since Cretaceous period that is from about 125 million years back. (Walker, 2009). The main reason for such dominating structure is due to continuous pollination. The flower pollination algorithm (FPA) is derived from the biological pollination process of flowers. The algorithm basically simulates flower constancy and pollination behaviour of flowering plants in nature. Pollination is typically performed by the transfer of pollen grain from one flower to another. This process can be biotic or abiotic. Biotic pollination follows transfer of pollens via insects or animals and about 90% of the flower species follow this pattern while rest of the 10% follow abiotic pollination in which no pollinators are required. Another phenomenon which adds to FPA is the flower constancy in which pollinators visit only particular species of flowers (Chittka, Thomson, & Waser, 1999). This process is advantageous for both the flower and the pollinator. The pollinator get sufficient amount of nectar whereas the flower maintains pollination and hence increase the population of particular flower species. Depending upon the availability of pollinators, pollination can also be defined as self and cross pollination. In self–pollination, there is no reliable pollinator while in cross–pollination, pollinators such as bee, birds and bats fly over long distances and results in global pollination (Pavlyukevich, 2007).

The pollination process, pollinator behaviour and flower constancy together sum up to form following four rules (Yang, 2012):

- The combination of biotic and cross-pollination is defined as global pollination and is done via Lèvy flights.
- Local pollination is performed by abiotic and self-pollination.
- The reproduction probability or the flower constancy is proportional to the similarity between two flowers.
- Another parameter switch probability $p \epsilon [0, 1]$ controls the local and global pollination.

These four rules are idealized to form mathematical equations. In order to simplify the procedure, only a single flower producing single pollen gamete is assumed as the potential solution of the problem under test. The algorithm was further formulated by exploiting the concept of local and global pollination. In global pollination, pollinators carry flowers over long distances. This helps in pollination and hence reproduction of the fittest (T^*)flower. Rule 1 in combination with flower constancy is represented as:

$$\mathrm{x}_{\mathrm{i}}^{\mathrm{t}+1} = \mathrm{x}_{\mathrm{i}}^{\mathrm{t}} + \mathrm{L}(\lambda)\left(\mathrm{T}^* - \mathrm{x}_{\mathrm{i}}^{\mathrm{t}}\right) \tag{1}$$

where x_i^t is the ith solution in the tth iteration, L corresponds to Lèvy flights used to generate random step size drawn from a standard Lèvy distribution as

$$L \sim \left\{ \frac{\lambda\Gamma\,(\lambda)\sin(\pi\lambda/2)}{\pi}\;\frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0) \right. \tag{2}$$

Here $\Gamma(\lambda)$ is the gamma function and $\lambda = 1.5$. The second phase is local pollination. This phase along with flower constancy are given by

$$x_i^{t+1} = x_i^t + \epsilon\left(x_j^t - x_k^t\right) \tag{3}$$

where ϵ is a uniformly distributed random number, x_j^{t+1} is the current solution, x_j^t and x_k^t are pollens or solution of same plant with different flower, mimicking flower constancy behaviour in a

Algorithm 1: Pseudo−code of flower pollination algorithm.

```
Begin:
Initialize flower population: $n$ with switch probability $p \epsilon [0, 1]$
Define $d$−dimensional objective function, $f(x)$
Find the initial best solution: $T*$
do Until iteration counter < maximum number of iterations
    for i = 1:n
        if rand < p
        A d−dimensional Lѐvy based step vector is drawn.
        Perform global Pollination: $x_i^{t+1} = x_i^t + L(\lambda)(T^* - x_i^t)$
        else
        perform local Pollination: $x_i^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t)$
        end if
        evaluate $x_i^{t+1}$
        if $x_i^{t+1}$ better than $x_i^t$, $x_i^t = x_i^{t+1}$
        end if
    end for
    Update current best.
    Update iteration counter
  end until
End
```

limited search space. It is further said that flower pollination occurs at local as well as global scale. Here Rule 4 above controls the pollination at local and global level. The parameter which controls this process is called as the switching probability denoted by p (Algorithm 1).

3. Literature

3.1. Review

The performance of any algorithm is based upon the phenomenon of exploration and exploitation. These two processes are also called as diversification and intensification respectively. The exploration process helps to perform global search that is to explore the unexplored areas of the search space where as exploitation is to perform local search in order to achieve more intensive results. It has been found in the literature (Črepinšek, Liu, & Mernik, 2013) that a good balance between both these phenomena is required to achieve better results. However, it is still a matter of concern and more research is to be done to interpret which part of meta-heuristic algorithm does exploration and which one is meant for exploitation (Draa, 2015).

Based on the above two phenomena, a brief over view of FPA is given in this section. FPA as described in Section 2, follow local and global pollination. It can be assumed from the above discussion itself that local pollination corresponds to the exploitation phase whereas global pollination is for exploration phase. There is another parameter named as switch probability (p) which decides the amount of exploration and exploitation. Here, if the value of p is 0.5, it means 50% exploration and 50% exploitation will take place. Likewise, if the value of p is [0.6 to 1], the rate of exploitation will increase and if it is decreased from 0.5, exploration will increase. In the proposed publication of FPA (Yang, 2012), it is said that one can use $p = 0.5$ as the initial value and further without any parametric study quoted that a value of 0.8 provides better results for most of the applications. Various other works on FPA in Sharawi et al. (2014), Prathiba et al. (2014), Singh, and Salgotra (2016), Yang, Karamanoglu, and He (2013), Pambudy, Hadi, and Ali (2014) and Ochoa et al. (2014) use same p value of 0.8 and no explanation regarding that has been given. In Draa (2015), an extensive survey about the values of p has been done qualitatively and it has been reported that $p = 0.2$ gives much better results as compared to $p = 0.8$. In Nabil (2016), a new modified FPA using clonal property has been proposed. The algorithm introduces a step-size scaling factor and it was said that a preliminary parametric study showed that a value of 3 works well for all test functions. But no such parametric study was given in the article. Also, the article uses $p = 0.8$ as standard and no detailed study about the selection of such value has been discussed.

For any generalized algorithm, we know that more global search should be done at the beginning as it will tend to explore the search space better and when enough exploration has been done, the process should move on to local search that is exploitation (Zhou, Wang, & Luo, 2016). Thus, use of dynamic switching will help to adjust the appropriate proportion of two searching processes. In FPA, local and global search are controlled by using switching probability and in Wang, and Zhou (2014), the concept of dynamic switching was formulated. Some limited work has also been done on the analysis of value of p FPA in Balasubramani and Marcus (2014), Sakib, Kabir, Subbir, and Alam (2014) and Łukasik and Kowalski (2015). It was concluded that no proper investigation about the value of p was given and no statistical testing of the results have been presented which further makes the analysis questionable.

Apart from parameter settings, a lot of modifications have been proposed in the recent past. Hybrid chaotic-based FPA extension was proposed to solve large integer problem (El-henawy & Ismail, 2014), DE hybridization was proposed and validated on benchmark function in Chakraborty, Saha, and Dutta (2014), a dimension-by-dimension approach was proposed by Wang, and Zhou (2014), flower pollination based maximum power point method for solar power tracking was proposed in Ram, and Rajasekar (2016), FPA with fuzzy approach proposed by Valenzuela, Valdez, and Melin (2017), a bat flower pollinator (BFP) was proposed in Salgotra and Singh (2016). The authors in Draa (2015) provided a proper quantitative and qualitative analysis of the vagueness of these proposed techniques. Further, they proposed five opposition based variants of FPA and tested their performance on standard (Liang, Qu, Suganthan, & Hernández-Díaz, 2013) and real world benchmarks (Suganthan et al., 2005). A deeper analysis of these algorithms has also been performed and it is shown that with respect to FPA they are better but overall, they have average performance as compared to state-of-the-art algorithms. Another recent modification includes elite opposition based FPA, which stressed on the global elite opposition, local self-adaptive greedy search and dynamic switch probability (Zhou et al., 2016). The dynamic switching helped to balance the exploration and exploitation while other helped to enhance the local and global pollination.

3.2. Inferences drawn from literature and motivation behind present work

The following inferences have been drawn from the literature:

- Proper switch probability p should be identified.
- In global pollination, the Lѐvy distribution based random number has not been explored in any of the work.
- Defining a better global and local search equation is required to make FPA a better state-of-the-art algorithm.

The inferences have motivated the authors to estimate that

- The switch probability p should be dynamic in nature in order to control the exploration and exploitation capability of FPA.
- Lѐvy distribution based random number can be treated same as mutation operator and must be exploited.
- In local pollination, the epsilon (ϵ) variable is to be analysed for some standard value to optimize the local search.

4. Proposed versions of flower pollination algorithm

With time, the problems being optimized are becoming more and more complex and so do the evolutionary algorithms. It has

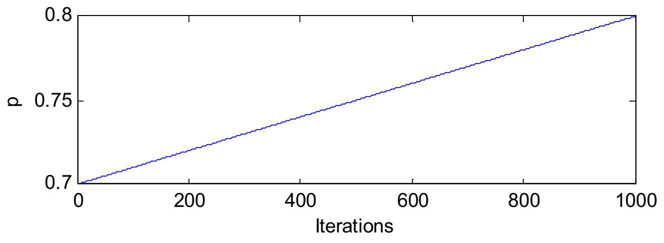

Fig. 2. Dynamic switch probability versus iterations.

been proved by Wolpert and Macready that no algorithm can be generic for all optimization problems (Wolpert & Macready, 1997). So, it becomes a necessity to optimize the algorithm by using appropriate mutation or selection operators. In this section, the concept of mutation is exploited and applied to standard FPA algorithm. Different variants are proposed and these variants are based upon the following basic ideas:

- A new concept based on mutation operators is applied to the global pollination phase.
- Based on mutation, five new variants of FPA are proposed.
- The first and the second variants exploit the concept of Cauchy and Gaussian distribution.
- The third and the fourth variants use a combination of Cauchy and Gaussian distribution in two different ways: firstly, by using mean of both and then by using an adaptive component of both respectively.
- The fifth one uses adaptive in both global as well as in local search.
- Dynamic switch probability is used in all the proposed variants.
- Analysis of FPA as an extension of DE.

Dynamic switch probability: As elaborated in the literature, switch probability plays a very important role on the performance of FPA and using it dynamically makes the algorithm more reliable in exploring and exploiting the search space. The general equation for dynamic switch probability p used here is given by:

$$p = p - \frac{maxiter - t}{maxiter} \times (0.1) \tag{4}$$

Here t is the current iteration and *maxiter* is the total number of iterations. The dynamic p value is used in all the proposed variants of FPA and so is not explicitly discussed in every subsection. The graph for variation in switching probability p with respect to iterations is shown in Fig. 2. The initial value of p is taken as 0.8.

4.1. Gaussian–FPA

The Gaussian mutation operation has been derived from the Gaussian normal distribution and its application to evolutionary search was formulated by Rechenberg and Schwefel (Bäck & Schwefel, 1993). This theory was referred to as classical evolutionary programming (CEP) (Yao, Liu, & Lin, 1999; Lee & Yao, 2001). The Gaussian mutations increase the searching capabilities of an algorithm (Rudolph, 1997) and has been used to exploit the desirable properties (Yao, Lin, & Liu, 1997). Also, Gaussian mutation is more likely to create a new offspring near the original parent because of its narrow tail. Due to this, the search equation will take smaller steps allowing for every corner of the search space to be explored in a much better way. Hence it is expected to provide relatively faster convergence. The Gaussian density function is given by:

$$f_{gaussian(0,\sigma^2)}(\alpha) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\alpha^2}{2\sigma^2}} \tag{5}$$

Algorithm 2: Pseudo–code of Gaussian – flower pollination algorithm (GFPA).

```
Begin:
  Initialize flower population: n
  Define dynamic switch probability
  Define d–dimensional objective function, f(x)
  Find the initial best solution: T*
  do Until iteration counter < maximum number of iterations
    for $i = 1{:}n$
      if $rand < p$
      A d–dimensional Gaussian based step vector is drawn.
      Perform global Pollination: $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{G}(\alpha)(\mathbf{T}^* - \mathbf{x}_i^t)$
      else
      set $\epsilon$ to be uniformly distributed in $[-1, 1]$
      perform local Pollination: $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \epsilon(\mathbf{x}_j^t - \mathbf{x}_k^t)$
      end if
      evaluate $x_i^{t+1}$
      if $x_i^{t+1}$ better than $x_i^t$, $x_i^t = x_i^{t+1}$
      end if
      Update switch probability: $\mathbf{p} = \mathbf{p} - \frac{\mathbf{maxiter-t}}{\mathbf{maxiter}} \times (\mathbf{0.1})$
    end for
    Update current best.
    Update iteration counter
  end until
End
```

where σ^2 is the variance for each member of the population. This function is further reduced to generate a single n–dimensional random variable by setting the value of mean to zero and standard deviation to 1. The random variable generated is applied to the general equation of global pollination phase of standard FPA algorithm and is given as

$$\mathrm{x}_i^{t+1} = \mathrm{x}_i^t + \mathrm{G}(\alpha)\left(\mathrm{T}^* - \mathrm{x}_i^t\right) \tag{6}$$

Here all the notations are same as that of FPA except for G(α), which corresponds to a d–dimensional Gaussian step vector generated using Gaussian density function with α as a Gaussian random number between [0, 1]. The pseudo–code for Gaussian–FPA (GFPA) is given in Algorithm 2.

4.2. Cauchy–FPA

Premature convergence can occur if relative minima of fitness functions are far away from small mutation of CEP. To overcome this effect, Yao *et al.* used a Cauchy distributed random number to generate a new solution (Yao et al., 1999). In FPA, the Lévy based step size mutation operator is switched to Cauchy based operator $C(\delta)$ to control smaller steps and the algorithm is named as Cauchy FPA (CFPA). This operator is simply a Cauchy based random number generated from Cauchy distribution function given by:

$$y = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{\delta}{g}\right) \tag{7}$$

The Cauchy density function is given by

$$f_{Cauchy(0,g)}(\delta) = \frac{1}{\pi}\frac{g}{g^2 + \delta^2} \tag{8}$$

The value of g is set to 1 and is the scale parameter, y, is a uniformly distributed random number between [0, 1]. Solving Eq. (7) for the value of δ, we get

$$\delta = \tan\left(\pi\left(y - \frac{1}{2}\right)\right) \tag{9}$$

This operator modifies the general equation in global pollination of the standard FPA to

$$x_i^{t+1} = x_i^t + C(\delta)\left(T^* - x_i^t\right) \tag{10}$$

Here use of Cauchy based mutation operator allows for larger mutation and hence helps in exploring the search space. This is because Cauchy distribution has a fatter tail and this tail helps to generate a larger step, which can help in escaping local minima and this is the reason for its enhanced performance. It further helps to avoid premature convergence. The exploitation in this case is handled by the local pollination Eq. (3) of the standard FPA. Due to the presence of Cauchy mutation operator, the algorithm is able to search more areas inside the search space which further makes it possible for the local pollination phase to exploit the available space in a much better way. The pseudo−code is same as that of GFPA except for change in the global pollination phase where Eq. (6) is changed to Eq. (10).

4.3. Combined mutated−FPA

Combined mutation refers to use of combination of two or more mutation operators. The concept of combined mutation was first formulated in Chellapilla (1998). Here they used a combination of Cauchy and Gaussian distributions, convolve them into a single distribution and based on that generate a new solution. The two combined mutation operators formulated by this process are known as mean mutated (MM) operator and adaptive mean mutation (AMM) operator (Koenig, 2002). The general equation for MM and AMM are given by

For MM:

$$x_i^{t+1} = x_i^t + 0.5\sigma'(\mathrm{G}(\alpha) + C(\delta)) \tag{11}$$

For AMM:

$$x_i^{t+1} = x_i^t + \sigma_1'\mathrm{G}(\alpha) + \sigma_2' C(\delta) \tag{12}$$

where all the notations are same as that used in previous subsections, σ_2' is a part of population and is identical to σ_1' making the mutations as:

$$\sigma_i' = \sigma_i \exp\left(r'.\mathrm{G}(\alpha) + r.G_t(\alpha)\right) \tag{13}$$

where $i = 1, 2, \ldots N$, for N population size, $\mathrm{G}(\alpha) and G_t(\alpha)$ are random numbers generated form Gaussian distribution with $G_t(\alpha)$ generated only once. The value of $r' and\ r$ are given by $r \propto (\sqrt{2\sqrt{n}})^{-1}$ and $r' \propto (\sqrt{2n})^{-1}$ respectively. Here the mean mutation (MM) and adaptive mean mutation (AMM) equations are not used as such. A simplified modification as per FPA is applied to the general equations and two new versions namely MM−FPA and AMM−FPA are proposed. The σ value generated here is a random solution for each member of the population group. The general equation of FPA uses the difference of two random solution, one generated from the best solution and other from any random solution in the population. So, for present case the σ value is changed as per FPA. The modification is applied in the global pollination phase and Eqs. (11) and (12) changes to:

For MM−FPA:

$$x_i^{t+1} = x_i^t + 0.5\left(T^* - x_i^t\right).(\mathrm{G}(\alpha) + C(\delta)) \tag{14}$$

For AMM−FPA:

$$x_i^{t+1} = x_i^t + \mathrm{G}(\alpha).\left(T^* - x_i^t\right) + C(\delta).\left(T^* - x_i^t\right) \tag{15}$$

The above procedure shows that MM uses sum of two distributions to generate new solution or offspring whereas AMM weighs each distribution before adding the two and hence allows the adaptation of not only distribution parameters, but also the shape. The pseudo−code for combined mutated FPAs is same as that given in Algorithm 2 except for the change in global pollination phase where Eq. (6) is replaced by Eq. (14) for MMFPA and (15) for AMMFPA.

4.4. Adaptive Lèvy−FPA

The concept of adaptive Lèvy was introduced by Lee and Yao (2001) in order to balance CEP with fine local and faster global search. This is done by deriving the fittest distribution among the candidate pool of four different distributions. The probability density function for Lèvy distribution is given by

$$f_{L\grave{e}vy(\lambda,\gamma)}(\lambda) = \frac{1}{\pi}\int_0^{\propto} e^{-\gamma q^{\lambda}}\cos(q\lambda)dq \tag{16}$$

where $1 < \lambda\langle 2, \gamma\rangle 0$ and is set to 1. On the basis of λ values, four different equations are proposed. When $\lambda = 1$, the distribution reduces to Cauchy and when $\lambda = 2$, it generates Gaussian distribution. Two other values of 1.3 and 1.7 are used to generate two Lèvy random 1 and Lèvy random 2 distributions respectively. The four equations are formulated using the values of λ are given as:

$$\mathrm{x}_{i,1}^{t+1} = \mathrm{x}_i^t + \mathrm{G}(\lambda)\left(\mathrm{T}^* - \mathrm{x}_i^t\right) \tag{17}$$

$$\mathrm{x}_{i,2}^{t+1} = \mathrm{x}_i^t + \mathrm{C}(\lambda)\left(\mathrm{T}^* - \mathrm{x}_i^t\right) \tag{18}$$

$$\mathrm{x}_{i,3}^{t+1} = \mathrm{x}_i^t + L_1(\lambda)\left(\mathrm{T}^* - \mathrm{x}_i^t\right) \tag{19}$$

$$\mathrm{x}_{i,4}^{t+1} = \mathrm{x}_i^t + L_2(\lambda)\left(\mathrm{T}^* - \mathrm{x}_i^t\right) \tag{20}$$

where $G(\lambda)$, $\mathrm{C}(\lambda)$, $L_1(\lambda)\,\text{and}\,L_1(\lambda)$ corresponds to values of Lèvy distributed random numbers with respect to Gaussian distribution, Cauchy distribution, Lèvy random 1 and Lèvy random 2 respectively. All the four equations are used simultaneously to generate new solutions and based upon the fitness one among them is selected. This increases the computational intensity of basic algorithm. By using the same phenomenon in the global pollination phase, adaptive Lèvy (AL) FPA is proposed in this section. A similar mechanism is followed in local pollination phase to counter balance the maximum number of search equations. The four search equations used in local pollination use two neighbouring solutions each and every solution is different. The general equations for this phase are given by

$$\mathrm{x}_{i,5}^{t+1} = \mathrm{x}_i^t + \epsilon\left(\mathrm{x}_j^t - \mathrm{x}_k^t\right) \tag{21}$$

$$\mathrm{x}_{i,6}^{t+1} = \mathrm{x}_i^t + \epsilon\left(\mathrm{x}_l^t - \mathrm{x}_m^t\right) \tag{22}$$

$$\mathrm{x}_{i,7}^{t+1} = \mathrm{x}_i^t + \epsilon\left(\mathrm{x}_n^t - \mathrm{x}_o^t\right) \tag{23}$$

$$\mathrm{x}_{i,8}^{t+1} = \mathrm{x}_i^t + \epsilon\left(\mathrm{x}_p^t - \mathrm{x}_q^t\right) \tag{24}$$

Here $\mathrm{x}_j^t, \mathrm{x}_k^t, \mathrm{x}_l^t, \mathrm{x}_m^t, \mathrm{x}_n^t, \mathrm{x}_o^t, \mathrm{x}_p^t$ and x_q^t corresponds to random solutions in the search pool with $j \neq k \neq l \neq m \neq n \neq o \neq p \neq q \neq i$ and each solution is generated from the combined population of flower pollinators. The value of ϵ is chosen from a uniform distribution in the range of [0, 1]. These four equations are evaluated based on their fitness and best among them is selected and used to generate the final output. Since the solution is updated four times, the population size is reduced to ¼ so that the number of function evaluations remains same with respect to standard algorithm. The pseudo−code for ALFPA is given in Algorithm 3.

4.5. FPA as an extension of DE

Since all the above proposed variants are based on the mutation operators so it is imperative to study the mutation operators of state-of-the-art algorithms with respect to mutation operators of FPA. The work here focuses on DE algorithm. The general discussion about basic DE is out of the scope of this paper and a better understanding about their basics can be have from publication (Storn & Price, 1997; Price, Storn, & Lampinen, 2006). First of all, let us discuss about the operators of DE with respect to FPA.

Algorithm 3: Pseudo−code of adaptive Lѐvy− flower pollination algorithm (ALFPA).

```
Begin:
  Initialize flower population: n/4
  Define dynamic switch probability
  Define d−dimensional objective function, f(x)
  Find the initial best solution: $T_*$
  do Until iteration counter < maximum number of iterations
    for $i = 1$:n
      if rand < p
        A d−dimensional Gaussian based step vector is drawn.
        perform global pollination: Generate $x_{i,1}^{t+1}$, $x_{i,2}^{t+1}$, $x_{i,3}^{t+1}$ and $x_{i,4}^{t+1}$
        evaluate each, find the best and replace it by $x_i^{t+1}$
      Else
        set $\epsilon$ to be uniformly distributed in [−1, 1]
        perform local pollination: $x_{i,5}^{t+1}$, $x_{i,6}^{t+1}$, $x_{i,7}^{t+1}$ and $x_{i,8}^{t+1}$
        evaluate each, find the best and replace it by $x_i^{t+1}$
      end if
      if $x_i^{t+1}$ better than $x_i^t$, $x_i^t = x_i^{t+1}$
      end if
      Update switch probability: $\mathbf{p} = \mathbf{p} - \frac{\mathbf{maxiter-t}}{\mathbf{maxiter}} \times (\mathbf{0.1})$
    end for
    Update current best
    Update iteration counter
  end until
End
```

DE has two main operators that is crossover rate (*CR*) and scaling factor (*F*). Various theories have been proposed claiming different values of *CR* and *F* to be efficient. Some quote it to be between $0.4 < F < 0.95$ and $0 < CR < 0.2$ (Ronkkonen, Kukkonen, & Price, 2005), some as $0.15 < F < 0.5$ and $0.1 < CR < 1.0$ (Omran, Salman, & Engelbrecht, 2005), some used fuzzy logics to define a value (Liu, & Lampinen, 2005), some use self−adaptive values (Qin, Huang, & Suganthan, 2009) and others use Gaussian distribution for F and self-adaptive values of *CR* (Abbass, 2002). So generally, the value of F and CR lies between [0, 1]. Now when we compare it with FPA, we see that *CR* and probability switch *p* both are basically the controlling parameters of exploration and exploitation for different algorithms whereas *F* and *L* are the scaling factors. Also in both the algorithms it has been explicitly proven that the values of *CR* and *p* either lies between [0, 1] or is self−adaptive or dynamic in nature and the values of *F* and *L* can be from any standard distribution function or it can also be a random variable. So, we can say *CR* is equivalent to *p* of FPA and *F* is simply the Lѐvy (L) distributed step size. Since the mutation parameters in both the algorithms are comparatively similar with values either chosen randomly or by using any specific distribution, so it is concluded that FPA is similar to DE. This conclusion is based on the fact that the general equation for DE is similar to Eq. (3) of FPA with only difference of *L* and *F*. *F* in DE is a random variable in the range of [0, 1] or a standard distribution. So, using *F* as Lѐvy distribution makes the general equation same as that of FPA. A proper exploitation phase is missing in the DE algorithm whereas in FPA improvement is added in terms of local pollination where the exploitation takes place. Overall, we can say that FPA is an extension of DE having scaling factor produced by a standard Lѐvy distribution and addition of one local pollination phase to improve the exploitation tendencies of DE.

5. Result and discussion

In this section, we provide details about the proposed variants of FPA, analyse them and see whether they improve the performance of basic FPA or not. First of all, we aim to study the proposed variants with respect to FPA for test functions and secondly compare the best variant among them with the state-of-the-art algorithms. The influence of population and mutation operators is also discussed explicitly.

For testing the variants, 50 runs are performed and following measures are used:

- Best of 50 runs
- Worst of 50 runs
- Mean of 50 runs
- Standard deviation of 50 runs
- Wilcoxon rank−sum test for statistical analysis.

5.1. PC configuration

The algorithms are run on Dell Inspiron 1464 having following configuration:

- Processor: Intel core i3, M330 @ 2.13 GHZ 2.13 GHz
- RAM: 4.00 GB
- OS: x64−bit, windows 10.
- MATLAB: 7.10.0 R2010a Version.

5.2. Test suite

The performance of all the proposed variants have been tested on 17 standard benchmark problems. These benchmark problems are uni-modal, multi-modal and fixed dimension functions. Uni-modal function has only one global optima and no local minima and hence are easy to solve. They help to estimate the convergence of an algorithm. Multi-modal functions on the other hand have many local minima and algorithms providing good results for them are capable of avoiding local minima. The dimensionality of uni-modal and multi-modal function depends on the user requirement and for present case we have taken 30 as the standard dimension for these functions. Third set of functions corresponds to the fixed dimension function, these functions mainly aim to show the consistency of an algorithm and are found to have fixed number of variables. The function along with their search range, dimension (D) and final optima are given in Table 1.

5.3. Influence of population

In the literature, it can be seen that work has been done to identify the influence of population on the performance of FPA algorithm (Draa, 2015) but the authors of that publication have changed the dimension as well. For example, if they are using the dimension of 10, the population size is also changed to 10 and if dimension is 20, the population size is also changed to 20. This makes it difficult to analyse the effect of population on the performance of FPA. In present work, the dimension of the problem is fixed and population size is varied. Three different population sizes (*n*) of 40, 60 and 80 are used to test the performance of FPA and the proposed variants. Table 2 shows comparison for $n = 40$, Table 3 for $n = 60$ and Table 4 for $n = 80$ for seventeen benchmark functions.

Case I: Population size 40: In Table 2, the simulation results for population size 40 are presented and it can be seen that for f_1, only ALFPA was able to reach global optimum while all other proposed versions stuck at some non-optimal solution. In f_2 ALFPA provides exact optimum results. For rest of the variants, the results are competitive but ALFPA is far better. For f_3, standard deviation for ALFPA is better in comparison to others. For f_4, ALFPA gives the worst results while all other variants fives quite competitive results. The best algorithm in this case is the CFPA. For f_5, AMMFPA gives the best results while others are still competitive. For f_6 and f_7 respectively, FPA and CFPA has the best standard deviation and are found to provide better results for these fixed dimensions function. For f_8 and f_9, AMMFPA, CFPA, GFPA, MMFPA and standard FPA are very competitive and among all AMMFPA is found to be best. For f_{10},

Table 1
Test functions.

Test problems	Objective function	Search range	Optimum value	D
Ackley function	$f_1(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp\ (\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20\ +\ \mathrm{e}$	[−100. 100]	0	30
Beale function	$f_2(x) = [1.5 - x_1(1-x_2)]^2 + [2.25 - x_1(1-x_2^2)]^2 + [2.625 - x_1(1-x_2^3)]^2$	[−4.5, 4.5]	0	2
Branin RCOS function	$f_3(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1-\frac{1}{8\pi})cosx_1 + 10$	$x_1 \in$ [−5, 10], $x_2 \in$ [0, 15]	0.397887	2
Expanded extended Griewank's plus Rosenbrock' s function (EEGR)	$f_4(x) =$ $f_{11}(f_{10}(x_1, x_2)) + f_{11}(f_{10}(x_2, x_3)) + \ldots + f_{11}(f_{10}(x_{D-i}, x_D)) + f_{11}(f_{10}(x_D, x_i))$	[−3, 1]	0	30
Griewank function	$f_5 = \frac{1}{4000}\sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N}\cos(\frac{x_i}{\sqrt{i}}) + \mathbf{1}$	[−600, 600]	0	30
Hartmann function 3	$f_6(x) = -\sum_{i=1}^{4}\alpha_i\exp[-\sum_{j=1}^{3}A_{ij}(x_j - P_{ij})^2]$	[0, 1]	−3.86278	3
Hartmann function 6	$f_7(x) = -\sum_{i=1}^{4}\alpha_i\exp[-\sum_{j=1}^{6}A_{ij}(x_j - P_{ij})^2]$	[0, 1]	−3.32237	6
Penalized 1 function	$f_8 = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)2[1\ +\ 10\sin2(\pi y_{i+1})_2\} + \sum_{i=1}^{n} u(x_i,\ 10,$ $100,\ 4)\ y_i = 1 + \frac{x_{i+1}}{4}$ $\mathrm{u}(x_i, \mathrm{a}, \mathrm{k}, \mathrm{m}) = \begin{cases} k(x_i - a)^m x_i > a \\ 0\ \ -a < x_i < a \\ k(-x_i - a)^m\ x_i < -a \end{cases}$	[−50, 50]	0	30
Penalized 2 function	$f_9 =$ $0.1\{(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + \mathbf{1})] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\} +$ $\sum_{i=1}^{n} u(x_i,\ 5,\ 100,\ 4)$ $\mathrm{u}(x_i, \mathrm{a}, \mathrm{k}, \mathrm{m}) = \begin{cases} k(x_i - a)^m x_i > a \\ 0\ \ -a < x_i < a \\ k(-x_i - a)^m\ x_i < -a \end{cases}$	[−50, 50]	0	30
Rastrigin function	$f_{10}(x) = 10D + \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i)]$	[−5.12, 5.12]	0	30
Rosenbrock function	$f_{11} = \sum_{i=1}^{N-1}(100(x_{i+1} - x_i^2)2 + (x_i - 1)2)$	[−100, 100]	0	30
Scaffer function	$f_{12}(x) = [\frac{1}{n-1}\sqrt{s_i}.(\sin(50.0s_i^{\frac{1}{5}}) + \mathbf{1})]^2 s_i = \sqrt{x_i^2 + x_{i+1}^2}$	[−100, 100]	0	30
Shekel function 5	$f_{13}(x) = -\sum_{j=1}^{5}[\sum_{i=1}^{4}((x_i - C_{ij})^2 + \beta_j)^{-1}]$	[0, 10]	−10.1532	4
Shekel function 7	$f_{14}(x) = -\sum_{j=1}^{7}[\sum_{i=1}^{4}((x_i - C_{ij})^2 + \beta_j)^{-1}]$	[0, 10]	−10.4029	4
Shekel function 10	$f_{15}(x) = -\sum_{j=1}^{10}[\sum_{i=1}^{4}((x_i - C_{ij})^2 + \beta_j)^{-1}]$	[0, 10]	−10.5364	4
Six Hump Camel function	$f_{16}(x) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	[−5, 5]	−1.0316	2
Easom function	$f_{17}(x) = -cosx_1 cosx_2 e^{(-(x_1-\pi)^2 - (x_2-\pi)^2)}$	[−10, 10]	−1	2

Table 2
FPA versus variants for population size of 40.

Function	Algorithm	Best	Worst	Mean	Standard deviation
f_1	ALFPA	**0.00373**	**19.9900**	**3.1356**	3.3740
	AMMFPA	20.9300	21.1399	21.036	0.0549
	CFPA	20.8745	21.1110	21.022	0.0560
	GFPA	20.9000	21.1521	21.053	0.0569
	MMFPA	20.5278	21.0611	20.968	0.0902
	FPA	20.8640	21.1032	21.016	0.0560
f_2	ALFPA	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	AMMFPA	3.63E−26	1.05E−16	6.37E−18	2.10E−17
	CFPA	1.12E−22	9.15E−15	1.86E−16	1.29E−15
	GFPA	4.22E−21	9.13E−13	4.76E−14	1.58E−13
	MMFPA	1.25E−23	5.05E−15	1.48E−16	7.21E−16
	FPA	6.47E−21	6.47E−15	5.27E−16	1.26E−15
f_3	ALFPA	0.3978	0.3970	0.3978	**3.36E−16**
	AMMFPA	0.3978	0.3970	0.3978	2.15E−08
	CFPA	0.3978	0.3978	0.3978	1.80E−07
	GFPA	0.3978	0.3978	0.3978	1.88E−07
	MMFPA	0.3978	0.3978	0.3978	6.55E−10
	FPA	0.3978	0.3978	0.3978	6.46E−08
f_4	ALFPA	6.05E+03	5.04E+06	6.58E+05	8.97E+05
	AMMFPA	8.0300	14.3300	11.703	1.4130
	CFPA	**7.5653**	15.4924	**11.600**	1.8970
	GFPA	15.360	21.6418	18.634	1.3218
	MMFPA	7.6993	13.5500	10.605	1.2262
	FPA	9.2850	18.1850	12.561	1.6038
f_5	ALFPA	2.6080	36.0340	12.619	8.1170
	AMMFPA	**1.0139**	1.13970	1.0540	0.0260
	CFPA	1.0214	1.11559	1.0503	0.0183
	GFPA	3.7351	14.9425	7.5357	2.4477
	MMFPA	1.1041	1.64950	1.2470	0.0989
	FPA	1.4316	4.98320	2.6520	0.7200
f_6	ALFPA	−3.3223	−3.2031	−3.2961	0.0498
	AMMFPA	−3.8620	−3.8620	−3.8620	2.90E−15
	CFPA	−3.8627	−3.8627	−3.8620	2.98E−15

(*continued on next page*)

Table 2 (*continued*)

Function	Algorithm	Best	Worst	Mean	Standard deviation
	GFPA	−3.8627	−3.8627	−3.8627	2.97E−14
	MMFPA	−3.8620	−3.8627	−3.8627	2.68E−15
	FPA	−3.8620	−3.8627	−3.8627	**2.62E−15**
***f*₇**	ALFPA	−3.3220	−3.2000	−3.2746	0.0589
	AMMFPA	−3.3220	−3.3220	−3.3223	2.16E−05
	CFPA	−3.3220	−3.3223	−3.3220	**2.76E−07**
	GFPA	−3.3223	−3.2031	−3.3156	0.0220
	MMFPA	−3.3223	−3.3223	−3.3223	8.05E−06
	FPA	−3.3223	−3.3223	−3.3223	5.93E−05
***f*₈**	ALFPA	8.7900	6.10E+06	2.41E+05	8.73E+05
	AMMFPA	**0.7958**	3.5535	1.7907	0.6150
	CFPA	0.8094	3.4541	1.6711	0.5568
	GFPA	3.9264	15.357	8.9485	3.0413
	MMFPA	1.2920	4.4990	2.2980	0.5545
	FPA	2.9310	7.6519	4.9427	1.2212
***f*₉**	ALFPA	4.37E+03	9.05E+06	8.37E+05	1.48E+06
	AMMFPA	**1.2290**	12.262	5.0250	2.5530
	CFPA	1.3755	9.6618	4.3841	2.0912
	GFPA	15.021	1.04E+04	5.39E+02	1.53E+03
	MMFPA	3.4262	24.1040	8.6310	3.9024
	FPA	11.552	75.2300	24.165	9.9719
***f*₁₀**	ALFPA	**32.795**	**84.7300**	**48.510**	**10.3373**
	AMMFPA	68.500	1.25E+02	93.788	13.7404
	CFPA	68.054	1.21E+02	97.5995	12.5135
	GFPA	1.68E+02	2.18E+02	1.93E+02	11.9709
	MMFPA	63.002	1.11E+02	84.0498	13.2060
	FPA	74.116	1.32E+02	97.7010	13.6325
***f*₁₁**	ALFPA	1.02E+07	3.07E+08	6.41E+07	6.70E+07
	AMMFPA	2.61E+03	**3.17E+04**	1.20E+04	7.16E+03
	CFPA	**2.28E+03**	3.48E+04	1.25E+04	8.77E+03
	GFPA	1.27E+06	1.945E+07	5.33E+06	3.84E+06
	MMFPA	8.96E+03	2.19E+05	5.82E+04	4.50E+04
	FPA	9.64E+04	4.12E+06	8.38E+05	8.51E+05
***f*₁₂**	ALFPA	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	AMMFPA	0.00E−00	1.538E−13	1.08E−14	2.45E−14
	CFPA	0.00E−00	5.74E−13	4.53E−14	1.24E−13
	GFPA	2.55E−14	7.10E−11	3.27E−12	1.07E−11
	MMFPA	0.00E−00	3.06E−11	1.66E−12	5.95E−12
	FPA	0.00E−00	7.50E−12	2.85E−13	1.10E−12
***f*₁₃**	ALFPA	−10.1531	−2.68286	−9.70497	1.7921
	AMMFPA	−10.1000	−10.1530	−10.1530	1.97E−08
	CFPA	−10.1530	−10.1530	−10.1500	**1.80E−08**
	GFPA	−10.1531	−10.1530	−10.1531	3.29E−05
	MMFPA	−10.1531	−10.1531	−10.1531	2.01E−07
	FPA	−10.153	−10.1531	−10.1530	8.82E−07
***f*₁₄**	ALFPA	−10.153	−2.6828	−9.90274	1.2630
	AMMFPA	−10.536	−10.5364	−10.5364	4.14E−07
	CFPA	−10.153	−10.1530	−10.1531	**1.72E−08**
	GFPA	−10.536	−10.5352	−10.5363	1.90E−04
	MMFPA	−10.536	−10.5363	−10.5360	9.00E−06
	FPA	−10.536	−10.5363	−10.5363	7.31E−06
***f*₁₅**	ALFPA	−10.530	−2.87114	−9.96175	1.9733
	AMMFPA	−10.530	−10.5300	−10.5300	5.81E−07
	CFPA	−10.536	−10.5360	−10.5360	**9.02E−08**
	GFPA	−10.536	−10.5360	−10.5300	8.34E−05
	MMFPA	−10.536	−10.5360	−10.5360	2.55E−06
	FPA	−10.536	−10.5360	−10.5364	8.01E−06
***f*₁₆**	ALFPA	−1.0316	−1.0316	−1.03162	**6.72E−16**
	AMMFPA	−1.0310	−1.0316	−1.03160	8.10E−15
	CFPA	−1.0316	−1.0316	−1.03162	6.50E−15
	GFPA	−1.0316	−1.0316	−1.03162	3.62E−12
	MMFPA	−1.0316	−1.0316	−1.03162	9.21E−14
	FPA	−1.0316	−1.0316	−1.03162	2.15E−12
***f*₁₇**	ALFPA	−1	−1	−1	**0.00E−00**
	AMMFPA	−1	−1	−1	**0.00E−00**
	CFPA	−1	−1	−1	**0.00E−00**
	GFPA	−1	−1	−1	5.40E−16
	MMFPA	−1	−1	−1	**0.00E−00**
	FPA	−1	−1	−1	**0.00E−00**

Table 3
FPA versus variants for population Size of 60.

Function	Algorithm	Best	Worst	Mean	Standard deviation
f_1	ALFPA	**0.0066**	**2.9669**	**1.3161**	0.9676
	AMMFPA	20.8448	21.0875	20.9943	0.0449
	CFPA	20.8232	21.1178	21.0025	0.0557
	GFPA	20.7637	21.1612	21.0286	0.0738
	MMFPA	20.8230	21.0813	20.9578	0.0555
	FPA	20.8609	21.1054	20.9924	0.0591
f_2	ALFPA	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	AMMFPA	5.83E−22	1.61E−17	1.11E−18	2.70E−18
	CFPA	1.77E−22	5.26E−17	2.20E−18	8.99E−18
	GFPA	1.43E−17	2.98E−12	1.651E−13	4.83E−13
	MMFPA	1.98E−20	9.94E−16	8.36E−17	1.88E−16
	FPA	8.43E−20	1.96E−14	1.11E−15	3.49E−15
f_3	ALFPA	0.3979	0.3979	0.3979	**3.36E−16**
	AMMFPA	0.3979	0.3979	0.3979	2.03E−10
	CFPA	0.3979	0.3979	0.3979	2.80E−10
	GFPA	0.3979	0.3979	0.3979	8.08E−09
	MMFPA	0.3979	0.3979	0.3979	3.03E−08
	FPA	0.3979	0.3979	0.3979	**6.33E−08**
f_4	ALFPA	4.1370	14.1461	9.3538	2.3602
	AMMFPA	8.5498	13.7908	11.6130	1.1440
	CFPA	7.9767	16.3025	12.5144	1.5910
	GFPA	17.4437	23.3525	20.0652	1.3255
	MMFPA	8.092	13.590	10.847	1.2510
	FPA	9.2407	15.7750	12.8020	1.2343
f_5	ALFPA	0.9780	3.0006	1.4037	**0.4652**
	AMMFPA	1.0301	1.1244	1.0682	0.0207
	CFPA	1.0387	1.1277	1.0656	0.0179
	GFPA	3.0325	10.8677	6.4368	1.5913
	MMFPA	1.157	1.58	1.309	0.0846
	FPA	1.7427	4.2795	2.6473	0.5529
f_6	ALFPA	−3.8628	−3.8628	−3.8628	**3.14E−15**
	AMMFPA	−3.8628	−3.8628	−3.8628	2.98E−15
	CFPA	−3.8628	−3.8628	−3.8628	2.90E−15
	GFPA	−3.8628	−3.8628	−3.8628	3.21E−14
	MMFPA	−3.8628	−3.8628	−3.8628	2.53E−15
	FPA	−3.8628	−3.8628	−3.8628	3.44E−15
f_7	ALFPA	−3.3224	−3.3224	−3.3224	**2.45E−08**
	AMMFPA	−3.3224	−3.3224	−3.3224	1.49E−07
	CFPA	− 3.3224	−3.3224	− 3.3224	2.42E−07
	GFPA	−3.3220	−3.3220	−3.3220	0.0019
	MMFPA	−3.3224	−3.3223	− 3.3224	4.71E−06
	FPA	−3.3224	−3.3220	−3.3223	7.17E−05
f_8	ALFPA	0.5598	1.31E+03	42.3065	1.90E+02
	AMMFPA	**0.4738**	2.7613	1.6745	0.5027
	CFPA	0.8623	3.1651	1.6862	**0.4850**
	GFPA	4.5005	19.3595	10.5529	2.9251
	MMFPA	1.6914	4.2418	2.5298	0.5655
	FPA	3.5667	7.8287	5.2567	1.1173
f_9	ALFPA	4.6284	2.76E+04	1.75E+03	4.77E+03
	AMMFPA	2.6780	8.6061	5.1385	**1.4795**
	CFPA	**2.1812**	8.3117	4.7075	**1.2923**
	GFPA	20.1167	6.09E+03	618.3793	1.07E+03
	MMFPA	4.4937	19.1556	9.6967	2.7398
	FPA	11.2756	37.5073	20.9544	5.6986
f_{10}	ALFPA	**22.0486**	**63.2669**	**42.6067**	11.2227
	AMMFPA	70.5701	125.3912	97.3845	13.6124
	CFPA	78.3628	129.0463	99.3214	10.1764
	GFPA	157.2383	231.0645	204.8374	14.0045
	MMFPA	56.7411	113.6353	90.1317	13.1490
	FPA	90.0512	134.3325	107.8787	10.1213
f_{11}	ALFPA	1.65E+04	9.24E+06	1.48E+06	2.11E+06
	AMMFPA	5.50E+03	4.10E+04	1.74E+04	**9.09E+03**
	CFPA	**4.94E+03**	7.80E+04	1.77E+04	1.26E+04
	GFPA	9.58E+05	1.19E+07	1.19E+07	2.86E+06
	MMFPA	1.39E+04	1.77E+05	7.65E+04	3.82E+04
	FPA	2.21E+05	1.69E+06	6.26E+05	3.25E+05
f_{12}	ALFPA	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	AMMFPA	0.00E−00	4.34E−13	3.84E−14	7.60E−14
	CFPA	0.00E−00	1.12E−12	4.83E−14	1.72E−13
	GFPA	4.44E−15	6.87E−10	2.17E−11	9.83E−11
	MMFPA	6.66E−16	2.11E−11	1.51E−12	3.67E−12
	FPA	0.000000	1.24E−12	1.97E−13	3.10E−13

(*continued on next page*)

Table 3 (*continued*)

Function	Algorithm	Best	Worst	Mean	Standard deviation
f_{13}	ALFPA	−10.1532	−10.1532	−10.1532	**8.97E−15**
	AMMFPA	−10.1532	−10.1532	−10.1532	7.35E−09
	CFPA	−10.1532	−10.1532	−10.1532	6.65E−09
	GFPA	−10.1532	−10.1532	−10.1532	2.30E−05
	MMFPA	−10.1532	−10.1532	−10.1532	5.50E−07
	FPA	−10.1532	−10.1532	−10.1532	1.81E−06
f_{14}	ALFPA	−10.4029	−10.4029	−10.4029	**8.97E−15**
	AMMFPA	−10.4029	−10.4029	−10.4029	6.52E−08
	CFPA	−10.4029	−10.4029	−10.4029	5.97E−08
	GFPA	−10.4029	−10.4034	−10.4012	3.18E−04
	MMFPA	−10.4029	−10.4029	−10.4029	3.68E−06
	FPA	−10.4029	−10.4029	−10.4029	3.42E−05
f_{15}	ALFPA	−10.5364	−10.5364	−10.5364	**8.93E−15**
	AMMFPA	−10.5364	−10.5364	−10.5364	1.00E−07
	CFPA	−10.5364	−10.5364	−10.5364	3.48E−08
	GFPA	−10.5364	−10.5363	−10.5363	1.92E−04
	MMFPA	−10.5364	−10.5364	−10.5364	1.86E−06
	FPA	−10.5364	−10.5364	−10.5364	1.26E−05
f_{16}	ALFPA	−1.0316	−1.0316	−1.0316	**6.72E−16**
	AMMFPA	−1.0316	−1.0316	−1.0316	3.61E−15
	CFPA	−1.0316	−1.0316	−1.0316	8.71E−16
	GFPA	−1.0316	−1.0316	−1.0316	3.53E−12
	MMFPA	−1.0316	−1.0316	−1.0316	7.81E−14
	FPA	−1.0316	−1.0316	−1.0316	4.54E−12
f_{17}	ALFPA	−1	−1	−1	**0.00E−00**
	AMMFPA	−1	−1	−1	**0.00E−00**
	CFPA	−1	−1	−1	**0.00E−00**
	GFPA	−1	−1	−1	9.64E−17
	MMFPA	−1	−1	−1	**0.00E−00**
	FPA	−1	−1	−1	**0.00E−00**

Table 4
FPA versus variants for population size of 80.

Function	Algorithm	Best	Worst	Mean	Standard deviation
f_1	ALFPA	**0.00626**	**2.14065**	**0.41643**	0.6685
	AMMFPA	20.8654	21.0827	20.9928	0.0423
	CFPA	20.8136	21.0619	20.9733	0.0567
	GFPA	20.8312	21.0996	21.0214	0.0564
	MMFPA	20.8157	21.0452	20.9444	0.0541
	FPA	20.8486	21.0715	20.9938	0.0465
f_2	ALFPA	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	AMMFPA	3.69E−21	4.74E−17	2.14E−18	6.94E−18
	CFPA	6.48E−22	2.53E−17	2.52E−18	6.02E−18
	GFPA	5.70E−17	6.18E−13	9.52E−14	1.60E−13
	MMFPA	3.63E−19	6.22E−16	9.19E−17	1.58E−16
	FPA	4.17E−18	1.11E−14	1.00E−15	1.91E−15
f_3	ALFPA	0.3979	0.3979	0.3979	**0.00E−00**
	AMMFPA	0.3979	0.3979	0.3979	3.14E−11
	CFPA	0.3979	0.3979	0.3979	4.70E−11
	GFPA	0.3979	0.3979	0.3979	1.28E−08
	MMFPA	0.3979	0.3979	0.3979	9.73E−10
	FPA	0.3979	0.3979	0.3979	4.54E−09
f_4	ALFPA	**5.52654**	**12.7332**	**8.44947**	1.6115
	AMMFPA	9.84276	15.2794	12.6325	1.3574
	CFPA	10.0031	16.4138	12.8217	1.4941
	GFPA	18.1922	24.1149	21.5050	1.4275
	MMFPA	7.1676 3	13.9665	11.2519	1.3002
	FPA	9.9236	16.8457	13.7463	1.3274
f_5	ALFPA	**0.0434**	**1.0897**	**0.5328**	0.2555
	AMMFPA	1.0566	1.2028	1.0904	0.0245
	CFPA	1.0457	1.1218	1.0846	0.0164
	GFPA	3.6287	4.0729	9.4301	2.4844
	MMFPA	1.2440	1.6534	1.3946	0.0863
	FPA	2.3040	4.3090	3.0930	0.4896
f_6	ALFPA	−3.8628	−3.8628	−3.8628	3.14E−15
	AMMFPA	−3.8628	−3.8628	−3.8628	2.90E−15
	CFPA	−3.8628	−3.8628	−3.8628	2.86E−15
	GFPA	−3.8628	−3.8628	−3.8628	7.83E−14
	MMFPA	−3.8628	−3.8628	−3.8628	2.45E−15
	FPA	−3.8628	−3.8628	−3.8628	**2.52E−15**

(*continued on next page*)

Table 4 (*continued*)

Function	Algorithm	Best	Worst	Mean	Standard deviation
f_7	ALFPA	−3.3224	−3.3224	−3.3224	0.0230
	AMMFPA	−3.3224	−3.2031	−3.3170	1.60E−07
	CFPA	**−3.3224**	**−3.3224**	**−3.3224**	**1.63E−07**
	GFPA	−3.3224	−.3224	−3.3224	0.0017
	MMFPA	−3.3224	−.3224	−3.3224	5.77E−06
	FPA	−3.3224	−3.3224	−3.3224	6.98E−05
f_8	ALFPA	**4.24E−04**	8.54	2.023	1.908
	AMMFPA	0.855	3.10	1.867	0.504
	CFPA	1.1781	3.0387	1.9842	0.4685
	GFPA	7.3902	44.8770	15.9797	6.0123
	MMFPA	1.5601	4.2341	2.8578	0.5523
	FPA	4.3020	10.259	6.7020	1.4720
f_9	ALFPA	**0.335**	47.131	14.658	11.019
	AMMFPA	3.886	10.229	6.418	1.549
	CFPA	3.4130	9.9119	6.1760	1.3344
	GFPA	41.2730	8.07E+04	7.40E+03	1.34E+04
	MMFPA	6.4905	7.6718	1.7207	2.6676
	FPA	12.514	45.990	26.605	7.5090
f_{10}	ALFPA	**25.658**	72.952	**49.319**	11.261
	AMMFPA	77.488	1.31E+02	1.01E+02	12.747
	CFPA	76.6478	125.5905	100.6522	11.6035
	GFPA	174.5894	253.5865	212.980	217.3889
	MMFPA	65.8232	21.5544	89.2597	10.7760
	FPA	73.813	1.34E+02	1.03E+02	12.237
f_{11}	ALFPA	**2.86E+02**	5.35E+04	**7.33E+03**	**9.49E+03**
	AMMFPA	1.00E+04	6.80E+04	2.43E+04	1.16E+04
	CFPA	8.28E+03	4.88E+04	2.55E+04	1.26E+04
	GFPA	2.61E+06	3.54E+07	1.24E+07	7.03E+06
	MMFPA	4.37E+04	2.35E+05	1.16E+05	4.56E+04
	FPA	3.43E+05	3.34E+06	1.18E+06	6.73E+05
f_{12}	ALFPA	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	AMMFPA	9.76E−15	9.88E−12	9.01E−13	1.67E−12
	CFPA	0.00E−00	3.27E−13	3.47E−14	6.55E−14
	GFPA	3.26E−14	5.65E−11	5.67E−12	1.16E−11
	MMFPA	2.22E−16	1.55E−11	1.44E−12	2.70E−12
	FPA	2.44E−15	1.69E−12	1.61E−13	3.01E−13
f_{13}	ALFPA	−10.1532	−10.1532	−10.1532	**8.93E−15**
	AMMFPA	−10.1532	−10.1532	−10.1532	8.74E−09
	CFPA	−10.1532	−10.1532	−10.1532	4.73E−09
	GFPA	−10.1532	−10.1529	−10.1532	5.65E−05
	MMFPA	−10.1532	−10.1532	−10.1532	3.28E−07
	FPA	−10.1532	−10.1532	−10.1532	1.09E−06
f_{14}	ALFPA	−10.4029	−10.4029	−10.4029	**8.97E−15**
	AMMFPA	−10.5364	−10.4029	−10.5364	5.85E−08
	CFPA	−10.5364	−10.4029	−10.5364	3.49E−08
	GFPA	−10.5364	−10.4029	−10.5364	1.18E−04
	MMFPA	−10.5364	−10.4138	−10.5363	3.02E−06
	FPA	−10.5364	−10.5364	−10.5364	1.35E−05
f_{15}	ALFPA	−10.5364	−10.5364	−10.5364	**8.90E−15**
	AMMFPA	−10.5364	−10.5364	−10.5364	3.9E−08
	CFPA	−10.5364	−10.5364	−10.5364	4.88E−08
	GFPA	−10.5364	−10.5364	−10.5364	1.09E−04
	MMFPA	−10.5364	−10.5358	−10.5363	2.50E−06
	FPA	−10.5364	−10.5364	−10.5364	2.22E−05
f_{16}	ALFPA	−1.0312	−1.0312	−1.0312	6.72E−16
	AMMFPA	−1.0312	−1.0312	−1.0312	**6.63E−16**
	CFPA	−1.0312	−1.0312	−1.0312	1.28E−15
	GFPA	−1.0312	−1.0312	−1.0312	2.68E−12
	MMFPA	−1.0312	−1.0312	−1.0312	5.25E−14
	FPA	−1.0312	−1.0312	−1.0312	7.84E−13
f_{17}	ALFPA	−1	−1	−1	**0.00E−00**
	AMMFPA	−1	−1	−1	**0.00E−00**
	CFPA	−1	−1	−1	**0.00E−00**
	GFPA	−1	−1	−1	1.89E−16
	MMFPA	−1	−1	−1	**0.00E−00**
	FPA	−1	−1	−1	**0.00E−00**

all the algorithms are competitive but ALFPA provides the best results. For f_{11}, AMMFPA, CFPA and MMFPA are found to be highly competitive while rest are comparable. For f_{12}, ALFPA provides exact global optimum solutions. AMMFPA and CFPA also provide the best solution but for worst, mean and standard deviation ALFPA is far better. For f_{13}, f_{14} and f_{15} CFPA is found to be better among all in terms of best, worst, mean as well as standard deviation. For f_{16}, the values of best, worst and mean are same for all the variants. Here ALFPA gives the best standard deviation. For f_{17}, ALFPA, AMMFPA, CFPA, MMFPA and FPA give the exact zero standard deviation giving no clue on which should be the best for this function. In this case, GFPA is found to be better for no function, MMFPA for two, FPA for three, AMMFPA for five, ALFPA for 7 and CFPA for eight functions. So, overall CFPA is found to be best among all the proposed variants for population size of 40.

Case II: Population size 60: Table 3 shows the comparison of various proposed variant of FPA with the standard one. Here for f_1, only ALFPA gives a near optimum solution. In f_2 ALFPA gives an exact zero optimal solution. The results are comparable among rest of the variants but ALFPA is far better. For f_3, standard deviation for ALFPA is best among all. For f_4 and f_5, all the variant gives competitive results but ALFPA provides the best results among all. For f_6 and f_7 respectively, ALFPA has the best standard deviation and is found to provide better results for these fixed dimensions function. For f_8 *and* f_9, AMMFPA and CFPA are quite competitive while rest are comparable. For f_{10}, ALFPA provides the best results. For f_{11}, AMMFPA and CFPA are found to be highly competitive. For f_{12}, ALFPA, AMMFPA, CFPA and FPA gives zero best but for worst, mean and standard deviation ALFPA is much better. For f_{13}, f_{14}, f_{15} *and* f_{16}, ALFPA is found to be better among all in terms of standard deviation for these fixed dimension functions. For f_{17}, ALFPA, AMMFPA, CFPA, MMFPA and FPA gives the exact zero standard deviation and no algorithm can be stated as the best one. In this case, MMFPA for one, FPA for two, AMMFPA for five, CFPA for eight and ALFPA for fourteen functions. So, overall ALFPA is found to be best among all the proposed variants for population size of 60.

Case III: Population size 80: The results for this case are given in Table 4. For f_1 and f_2 ALFPA is able to attain global optimum while others are not. For f_3, ALFPA has the best standard deviation among all. For f_4 *and* f_5, all the variant gives competitive results but best values are obtained by ALFPA. For f_6, all variants have very competitive standard deviation and for f_7 CFPA gives the best results. For f_8, f_9, f_{10}, f_{11} and f_{12}, ALFPA gives the best results. For fixed dimension functions f_{13}, f_{14}, f_{15} again ALFPA gives the best results and for f_{16} AMMFPA and ALFPA both have almost same standard deviation. For f_{17}, ALFPA, AMMFPA, CFPA, MMFPA and FPA gives the exact zero standard deviation and no algorithm can be stated as the best one. In this case, GFPA for one, FPA and MMFPA for two, AMMFPA and CFPA for three and ALFPA for fifteen functions. So, overall ALFPA is found to be best among all the proposed variants for population size of 80.

Inferences drawn: The first thing to see here is that five new variants are proposed here and apart from GFPA, all other variants are found to be better than FPA algorithm. For population size of 40, FPA is able to provide better results for only three functions while for rest of the fourteen functions, the proposed variants are better. For population size of 60, FPA is better for only two and for 80 population size, it is better for two. So overall, among 51 functions, FPA gives better results for only seven functions while for rest of the function the proposed variants provide better results.

Among all the proposed variants, the worst algorithm is GFPA which is better for only one function, MMFPA is the second last and is better only for five functions, AMMFPA for thirteen functions, CFPA for sixteen functions and ALFPA for thirty-six functions. The total number of functions are only 51 for all the three cases combined but the results are for 78 functions. This is because there are some cases in which two or more variants gives exact result for a particular function. That is, for the function in which two algorithms give same results can be counted two times. The discussion shows that ALFPA gives the best performance among all the proposed variants.

Now as far as population size is concerned, it can be seen that as the population size increases from 40 to 60, the performance of ALFPA improves and it almost gets stagnated on further increase in population size. It can be said from the fact that for first case, ALFPA was better for seven functions while CFPA was better for eight but as the population size was increased in second case, ALFPA became better for fourteen test functions while others were far less comparable. In the final case of 80 population size, ALFPA became better for fifteen functions but the increase from previous case was only modest whereas the total number of function evaluations increased. Overall, we can assume a population size of 60 to be the standard population size for proposed variants of FPA.

Table 5
parameter settings for various algorithms.

Algorithm	Parameters	Values
ABC	Colony size	60
	Number of food sources	Colony size/2
	Limit	100
	Maximum cycles	1000
	Stopping criteria	Max iteration.
GWO	$\vec{a}$	Linearly decreased from 2 to 0.
	Population size	60
	Maximum number of generations	1000
	Stopping criteria	Max iteration.
FA	Number of fireflies	60
	Alpha (α)	0.5
	Beta (β)	0.2
	Gamma (γ)	1
	Maximum iterations	1000
	Stopping criteria	Max iteration.
DE	Population size	60
	F	0.5
	CR	0.5
	Maximum iterations	1000
	Stopping criteria	Max iteration.
BA	Population size	60
	Loudness	0.5
	Pulse rate	0.5
	Maximum number iterations	1000
	Stopping criteria	Max iteration.
ALFPA	Population size	60
	Probability switch	Dynamic
	Maximum iterations	1000
	Stopping criteria	Max iteration.

5.4. Comparative study

From the above discussion, it has been found that ALFPA is better than other FPA variants. From the inferences in Section 5.3, it can be said that the perfect population size for comparison should be 60 where the best algorithm is ALFPA. The reason for better performance of ALFPA at a population size of 60 is that for higher population size, the diversity in the population increases. Due to the increased diversity, the exploration increases helping the algorithm to search for better solutions and avoid local minima. This concept is also valid for exploitation or the local pollination phase. The same algorithm doesn't work well for lower population sizes (40) because of less diversity in the solution space. The population is divided into four parts and there are four equations working together to find global minima, so more is the population size more are the chances to get the required results. For higher population sizes, the algorithm gives good results, but it can be seen in Section 5.3 that ALFPA results don't vary too much with further increase in the population size. Now to prove it to be state-of-the-art, it is compared with well-known algorithms. The major algorithms used in this paper are ABC, DE, BA, FA and GWO and these algorithms are compared with the best proposed variant of FPA. Here all the popular algorithms are compared with ALFPA for a population size of 60. The parameters settings for these algorithms are given in Table 5.

The results are presented in Table 6 as: For function f_1, only ALFPA was able to reach global optimum. In f_2 ABC and ALFPA provides exact optimum results. DE is also able to obtain a global best of zero but for worst, mean and standard deviation the results are less significant. For rest of the algorithms, the results are compet-

Table 6
Results comparison for various algorithms.

Function	Algorithm	Best	Worst	Mean	Standard deviation
f_1	ABC	20.0000	20.0120	20.0032	1.26E−02
	DE	20.0000	20.0000	20.0000	3.02E−09
	BA	19.9990	20.0000	19.9999	2.22E−04
	GWO	20.5887	20.9715	21.8017	9.21E−02
	FA	19.9976	20.0023	20.0007	6.97E−04
	ALFPA	**6.60E−03**	**2.96E−00**	**1.31E−00**	9.67E−01
f_2	ABC	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	DE	0.00E−00	1.02E−01	2.00E−03	1.44E−02
	BA	6.24E−12	6.52E−01	9.50E−02	2.21E−01
	GWO	2.44E−10	7.62E−01	1.52E−02	1.07E−01
	FA	1.52E−12	5.81E−10	1.13E−10	1.02E−10
	ALFPA	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
f_3	ABC	0.00E−00	0.00E−00	0.00E−00	0.00E−00
	DE	0.3979	0.5735	0.4058	2.68E−02
	BA	0.3979	0.3979	0.3979	2.08E−10
	GWO	0.3979	0.3979	0.3979	3.24E−05
	FA	0.3978	0.3978	0.3978	1.24E−09
	ALFPA	0.3979	0.3979	0.3979	**3.36E−16**
f_4	ABC	**2.97E−00**	**8.32E−00**	5.48E−00	2.38E−00
	DE	2.60E+01	4.27E+02	1.14E+02	9.03E+01
	BA	1.27E+01	2.32E+02	6.46E+01	4.59E+01
	GWO	1.10E−00	1.99E+01	5.79E−00	4.38E−00
	FA	2.60E−00	1.00E+01	4.96E−00	1.37E−00
	ALFPA	4.13E−00	1.41E+01	9.35E−00	**2.36E−00**
f_5	ABC	7.38E−01	1.06E−00	8.84E−01	8.01E−02
	DE	4.67E+01	3.03E+02	1.36E+02	5.69E+01
	BA	6.26E+01	2.00E+02	1.20E+02	3.40E+01
	GWO	**0.00E−00**	1.34E−02	1.20E−03	3.40E−03
	FA	2.63E−03	6.40E−03	4.20E−03	1.00E−03
	ALFPA	9.78E−01	3.00E−00	1.40E−00	4.65E−01
f_6	ABC	−3.8628	−3.8627	−3.8628	1.67E−05
	DE	−3.8628	−3.8549	−3.8603	3.50E−02
	BA	−3.8628	−3.8628	−3.8628	1.82E−08
	GWO	−3.8628	−3.8549	−3.8624	1.70E−03
	FA	−3.8627	−3.8627	−3.8627	4.04E−10
	ALFPA	**−3.8628**	**−3.8628**	**−3.8628**	**3.14E−15**
f_7	ABC	−3.3223	−3.2030	−3.3148	2.85E−02
	DE	−3.3224	−2.8657	−3.1628	7.07E−02
	BA	−3.3224	−3.2032	−3.2842	5.62E−02
	GWO	−3.3224	−3.1327	−3.2752	6.48E−02
	FA	−3.3223	−3.1666	−3.2612	6.19E−02
	ALFPA	**−3.3224**	**−3.3224**	**−3.3224**	**2.45E−08**
f_8	ABC	1.56E−02	1.02E−00	1.87E−01	5.01E+07
	DE	3.24E+06	6.54E+08	1.85E+08	1.94E+08
	BA	8.51E+00	7.72E+06	8.21E+05	1.41E+07
	GWO	8.43E−07	7.19E−02	2.35E−02	1.68E−02
	FA	1.23E−05	**6.82E−05**	**2.55E−05**	**1.00E−05**
	ALFPA	5.59E−01	1.32E+03	4.23E+01	1.90E+02
f_9	ABC	1.35E−01	3.07E−00	8.32E−01	6.10E−01
	DE	3.54E+07	1.39E+09	3.19E+08	3.35E+08
	BA	6.37E+04	5.64E+07	9.39E+06	1.20E+07
	GWO	9.63E−06	6.86E−01	2.67E−01	1.93E−01
	FA	1.10E−04	**1.14E−02**	**5.82E−04**	**1.50E−03**
	ALFPA	4.62E−00	2.76E+04	1.75E+03	4.77E+03
f_{10}	ABC	6.06E−00	3.07E+01	1.44E+01	6.04E−00
	DE	9.87E+01	2.14E+02	1.53E+02	2.55E+01
	BA	3.18E+01	2.08E+02	7.45E+01	2.69E+01
	GWO	**0.00E−00**	**4.50E−00**	**1.10E−01**	**6.50E−01**
	FA	1.79E+01	6.16E+01	3.44E+01	1.01E+01
	ALFPA	2.20E+01	6.32E+01	4.26E+01	1.12E+01
f_{11}	ABC	1.35E+03	1.23E+05	2.02E+04	2.41E+04
	DE	1.19E+09	3.73E+10	8.21E+09	8.89E+09
	BA	8.51E+07	4.76E+09	1.16E+09	1.09E+09
	GWO	**2.51E+01**	**2.84E+01**	**2.63E+01**	**7.20E−01**
	FA	2.81E+01	7.28E+03	2.32E+03	2.57E+03
	ALFPA	1.65E+04	9.24E+06	1.48E+06	2.11E+06
f_{12}	ABC	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	DE	0.00E−00	1.51E−01	1.93E−02	3.37E−02
	BA	7.77E−05	1.95E−01	4.17E−02	5.38E−02
	GWO	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	FA	2.77E−13	2.13E−02	2.30E−03	4.30E−03
	ALFPA	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
f_{13}	ABC	−10.1525	−5.0589	−9.8957	9.97E−01
	DE	−10.1532	−1.8513	−5.3191	2.96E−00

(*continued on next page*)

Table 6 (*continued*)

Function	Algorithm	Best	Worst	Mean	Standard deviation
	BA	−10.1532	−2.6305	−6.2930	3.26E−00
	GWO	−10.1532	−5.0552	−9.6468	1.53E−00
	FA	−10.1531	−2.6828	−9.8543	1.47E−00
	ALFPA	**−10.1532**	**−10.1532**	**−10.1532**	**8.97E−15**
***f*₁₄**	ABC	−10.5056	−5.1516	−10.2474	1.05E−00
	DE	−10.5064	−2.8027	−9.3903	2.32E−00
	BA	−10.5048	−1.6766	−5.6230	3.63E−00
	GWO	−10.5049	−2.4217	−10.1585	7.57E−04
	FA	−10.4028	−10.4026	−10.4022	8.30E−07
	ALFPA	**−10.4028**	**−10.4028**	**−10.4028**	**8.95E−15**
***f*₁₅**	ABC	−10.5348	−5.1651	−10.1934	1.10E−00
	DE	−10.5364	−2.1824	−9.6726	2.18E−00
	BA	−10.5364	−1.8595	−5.1735	3.47E−00
	GWO	−10.5364	−10.5356	−10.5361	1.88E−04
	FA	−10.5364	−10.5364	−10.5364	7.30E−07
	ALFPA	**−10.5364**	**−10.5364**	**−10.5364**	**8.93E−15**
***f*₁₆**	ABC	−1.0316	−1.0315	−1.0316	3.77E−05
	DE	−1.0316	−1.0316	−1.0316	6.72E−16
	BA	−1.0316	−0.2155	−1.0153	1.15E−01
	GWO	−1.0316	−1.0316	−1.0316	3.99E−09
	FA	−1.0316	−1.0316	−1.0316	1.04E−10
	ALFPA	−1.0316	−1.0316	−1.0316	**6.72E−16**
***f*₁₇**	ABC	−1.0000	−0.9996	−0.9996	1.80E−01
	DE	−1.0000	−0.9702	−0.9988	5.70E−03
	BA	−1.0000	−1.0000	−1.0000	5.92E−11
	GWO	−1.0000	−1.0000	−1.0000	8.71E−08
	FA	−1.0000	−1.0000	−1.0000	1.36E−09
	ALFPA	−1.0000	−1.0000	−1.0000	**0.00E−00**

itive but ALFPA is still better. For f_3, ABC is not able to attain the required optimum, while all other algorithms provide competitive results. ALFPA in this case has the best standard deviation. For f_4, ABC, FA and ALFPA are competitive and better than GWO, BA and DE. Here it is difficult to comment whether ABC or ALFPA is better. For f_5, ALFPA is better than ABC, DE and BA whereas FA is competitive with respect to ALFPA. GWO is found to be the best. For f_6 *and* f_7, ALFPA has the best standard deviation is found to provide better results for these fixed dimensions function. For f_8 *and* f_9, ALFPA is better than DE and BA. GWO provides the best optimum but the overall results of FA are better. For f_{10}, ALFPA is very competitive when compared to ABC, DE, FA and BA but in this case GWO provides the best results. For f_{11}, ALFPA is better than DE and BA, very competitive with ABC and GWO is found to be the best. For f_{12}, ABC, GWO and ALFPA provide exact global optimum solutions. For f_{13}, f_{14} and f_{15} ALFPA is found to be better among all in terms of best, worst, mean as well as standard deviation. For f_{16}, the values of best, worst and mean are same for almost every algorithm. Here standard deviation gives the relative comparison among the algorithms. DE and ALFPA are found to provide the best competitive results. For f_{17}, also the standard deviation acts as the deciding factor. Here ALFPA gives the exact zero standard deviation and hence proving to be best among all. Overall, GWO and FA are found to be better for four functions and two functions each, DE for only one function, ABC for two functions while ALFPA gives best results for twelve functions. The convergence profiles are given in Fig. 3. These convergence curves again show the superior performance of AFLPA as compared to other algorithms. This conclusion proves the competitiveness of ALFPA and it can be said that ALFPA is best fit candidate for becoming state of art algorithm.

5.5. Statistical testing

The Wilcoxon's rank-sum test (Derrac, García, Molina, & Herrera, 2011) is performed to test the performance of ALFPA with other popular algorithms. This test is done to test the performance of two different algorithms or samples. It is a non−parametric test, involving design of two samples and is analogous to paired t-test. So, it is a pairwise test used to identify significant differences among two algorithms. The test gives a *p*-value as the end result and this *p*-value determines the significance level of two algorithms under test. An algorithm is said to be statistically significant if the *p*-value is $<$ 0.05. For comparison, two algorithms are taken for each test function and comparison is performed. This comparison is done with respect to the best algorithm that is if ALFPA is best, comparison is performed between ALFPA/ABC, ALFPA/DE, ALFPA/BA, ALFPA/GWO and so on. Since best algorithm cannot be compared with itself so NA has been incorporated for each algorithm and it stands for Not Applicable Saremi, Mirjalili & Lewis, 2017 while '∼' means that both the algorithms have same performance. From the Table 7, it can be seen that ABC shows better statistical analysis for two functions, DE and BA for none, GWO and FA for three each and ALFPA for twelve functions. So even from the statistical tests, it can be said that ALFPA is much better and is best fit for becoming a standard algorithm.

5.6. Summary of results

The main findings can be summarized as follows.

- The basic FPA is limited in performance and the already proposed work does not provide proper justification. The variants proposed in present work are competitive and provide good results. ALFPA is found to be the best among the proposed variants.
- The reason for better performance of ALFPA is because of the good exploration capabilities of adaptive Lèvy component in the global pollination phase, and, at the same time, it inherits the good exploitative capabilities of enhanced local search. Also, the dynamic switching helps to maintain a balance between exploration and exploitation. Overall, ALFPA combines three efficient tools for searching the global solution.
- Although ALFPA is found to be the best algorithm and acts as potential candidate for becoming a state-of-the art, it has some limitations too. The algorithm does not work well for lower population and this is because at lower population the algo-

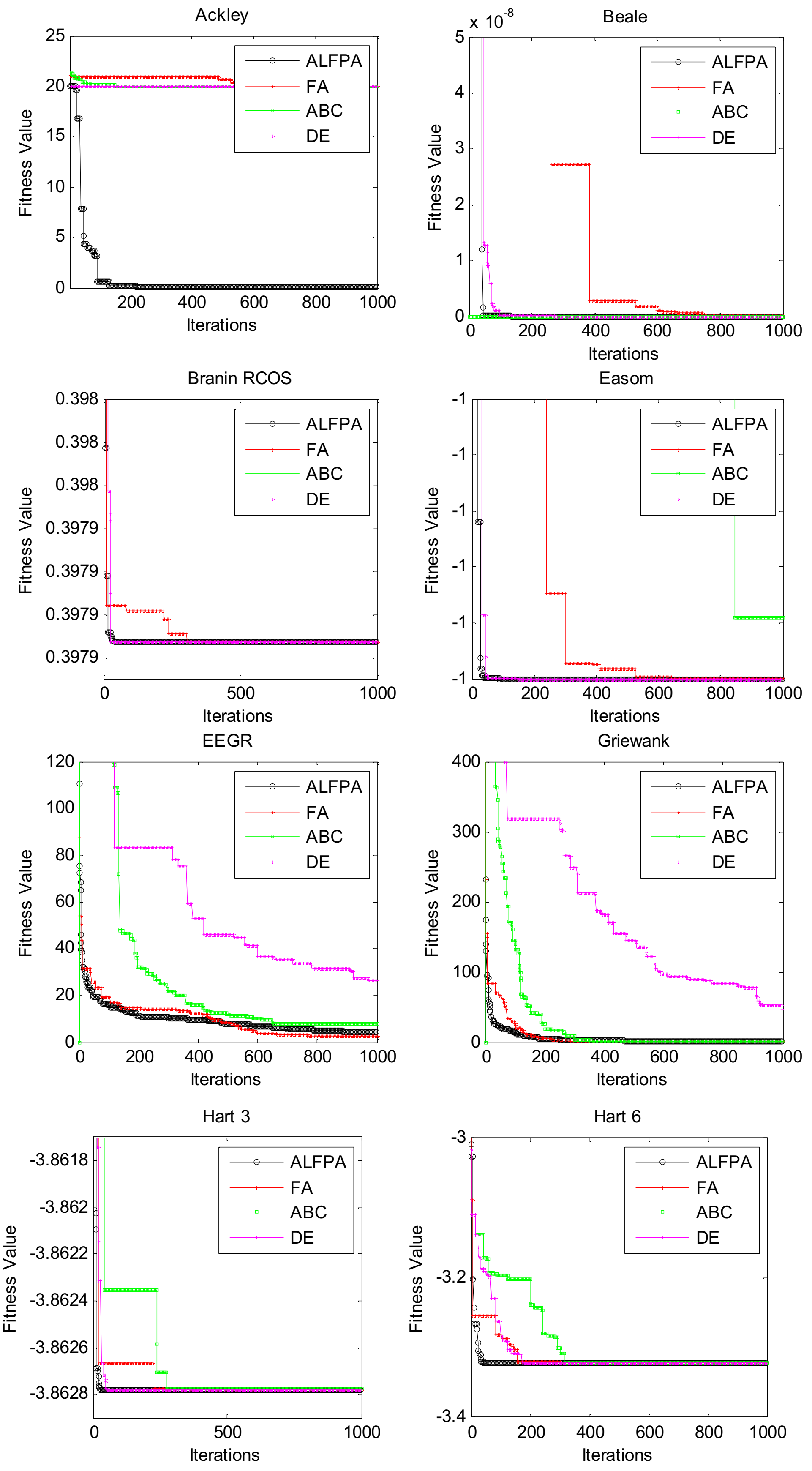

Fig. 3. Convergence profiles of various algorithms.

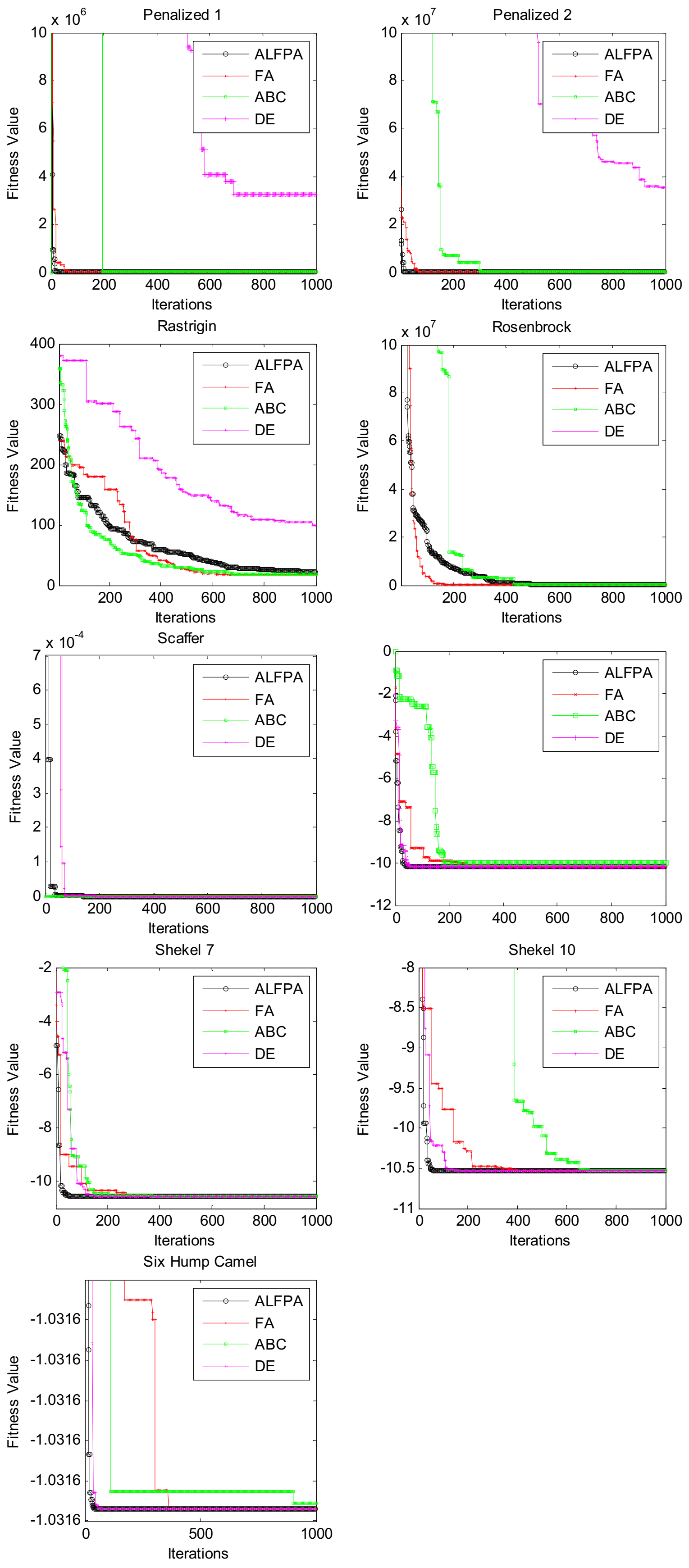

Fig. 3. Continued

Table 7
p-test values of various algorithms.

Function	ABC	DE	BA	GWO	FA	ALFPA
f_1	7.06E−18	5.18E−14	7.06E−18	~	7.06E−18	NA
f_2	~	8.34E−18	3.31E−20	3.31E−20	3.31E−20	NA
f_3	2.62E−23	1.89E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_4	2.27E−13	6.25E−19	9.54E−18	5.53E−15	5.53E−15	NA
f_5	7.06E−18	3.31E−20	7.06E−18	7.72E−12	NA	7.06E−18
f_6	3.31E−20	3.05E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_7	6.90E−16	6.90E−16	6.47E−17	1.20E−17	1.20E−17	NA
f_8	7.06E−18	3.31E−20	7.06E−18	2.29E−15	NA	7.06E−18
f_9	7.06E−18	3.31E−20	7.06E−18	5.92E−11	NA	7.06E−18
f_{10}	2.13E−19	4.18E−20	2.13E−19	NA	2.13E−19	2.13E−19
f_{11}	7.06E−18	3.31E−20	7.06E−18	NA	7.06E−18	7.06E−18
f_{12}	~	2.40E−20	3.31E−20	~	3.31E−20	NA
f_{13}	3.31E−20	3.04E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{14}	3.31E−20	1.87E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{15}	3.31E−20	2.33E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{16}	3.31E−20	2.74E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{17}	3.31E−20	3.31E−20	3.31E−20	3.31E−20	3.31E−20	NA

rithm diversity is limited. As the population size required for this algorithm is large, more is the number of function evaluations required to obtain a particular solution for the problem under test. So, more work is to be done to design FPA variant which is fit for lower population size.
- Due to stochastic nature of ALFPA and all other nature inspired algorithms, there is no guarantee for finding 100% optimal solution. Also, the proposed variant is not fit for all the optimization problems. That is ALFPA did not perform well on all benchmark functions. So, to overcome this problem more work is to be done to design a generic problem solver.
- Though, ALFPA has some limitation but still it fares better than the already proposed work in the literature. Also, the algorithm is quite simple in implementation and because of this it can be included in hybrid intelligent and expert systems.

Further, insightful implications extracted from the experiments are

- Parallelism of ALFPA can be done by dividing the search space into smaller parts using corresponding equations, then each equation under each search space can be used independently and these spaces can be made to interact with each other after certain set of iterations.
- Extending the algorithm to binary version can be used in electroencephalogram (EEG). In EEG, signals are measured by placing sensors in different positions on a person's head. A binary version of ALFPA can help to reduce total number of sensors required for the operation while maintaining the accuracy of the system.

6. Conclusion and future scope

This paper presents modifications to basic FPA in order to enhance its performance. Three modifications to FPA have been proposed which make it competitive with respect to popular algorithms. A dynamic switch probability, an improved mutated global and local search has been used in the enhanced versions of FPA. The mutation techniques give five variants of FPA in which Cauchy, Gaussian, adaptive Lévy, mean mutated and adaptive mean mutated distributions have been used. It has been found that all of these variants except GFPA are better than the basic FPA and ALFPA is best among all the variants. The performance of ALFPA has been also compared with that of well-known algorithms such as DE, BA, FA, ABC and GWO. Experimentally, best, worst, mean and standard deviation has been used to measure the results. Due to stochastic nature of optimization algorithms, statistical tests have been conducted. To measure the speed of attaining the global optima, convergence profiles have also been drawn. Numerical results show the strength of ALFPA in solving the standard benchmark problems.

The enhanced performance of variants can be attributed to dynamic switching which balances effectively local and global search. The global mutations also help in improving the exploration capabilities of the basic FPA which is required to escape local minima. In addition to these changes, in ALFPA the local search equation has been changed to improve the exploitation. The proposed methods of FPA have been tested for different population sizes and it has been observed that for lower population size CFPA gives better results and for medium population size ($n = 60$) ALFPA is better. Further if the population is increased, there is no significant improvement in the results. The CFPA has sufficient number of candidate solutions to explore the search space. Hence it is good at both exploration and exploitation. But in case of ALFPA for lower population size the diversity of possible solutions is less and hence its exploration is poor. Overall it has been seen that ALFPA fares better than all the variants and other popular algorithms.

ALFPA has proved its significance for solving benchmark problems but for it to be state-of -the-art, it has to be applied to various engineering optimization problems like antenna, wireless sensor networks, digital filter design problems, economic load dispatch problems, knapsack problem. It is also possible to study the extension of ALFPA for discrete combinatorial optimization problems. The switching probability which acts as the controlling factor for exploration and exploitation is very delicate. The dynamic switching probability used in proposed approaches is linear in nature. One can use sigmoidal or exponential functions to properly identify the best switching probability. A balanced exploration and exploitation can also be introduced. Further, one can extend the work by modifying the search equations of local and global pollination phase. More research can be done by exploiting the epsilon parameter in the local search part of FPA. The inertia weight factors can also be introduced in local and global pollination phase of standard FPA. Besides these implementations, step sizes can be analysed to propose new FPA variants.

References

Abbass, H. A. (2002, May). The self-adaptive pareto differential evolution algorithm. In *Evolutionary computation, 2002. CEC'02. Proceedings of the 2002 congress on: Vol. 1* (pp. 831–836). IEEE.

Bäck, T., & Schwefel, H. P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation, 1*(1), 1–23.

Balasubramani, K., & Marcus, K. (2014). A study on flower pollination algorithm and its applications. *International Journal of Application or Innovation in Engineering and Management, 3*, 230–235.

Chakraborty, D., Saha, S., & Dutta, O. (2014). DE-FPA: A hybrid differential evolution-flower pollination algorithm for function minimization. In *High performance computing and applications (ICHPCA), 2014 international conference on* (pp. 1–6). IEEE.

Chellapilla, K. (1998). Combining mutation operators in evolutionary programming. *IEEE Transactions on Evolutionary Computation, 2*(3), 91–96.

Chittka, L., Thomson, J. D., & Waser, N. M. (1999). Flower constancy, insect psychology, and plant evolution. *Die Naturwissenschaften, 86*(8), 361–377.

Črepinšek, M., Liu, S. H., & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR), 45*(3), 35.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation, 1*(1), 3–18.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine, 1*, 28–39.

Draa, A. (2015). On the performances of the flower pollination algorithm–Qualitative and quantitative analyses. *Applied Soft Computing, 34*, 349–371.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science: Vol. 1* (pp. 39–43).

El-henawy, I., & Ismail, M. (2014). An improved chaotic flower pollination algorithm for solving large integer programming problems. *International Journal of Digital Content Technology & its Applications, 8*(3).

Gutjahr, W. J. (2009). Convergence analysis of metaheuristics. In *Matheuristics* (pp. 159–187). Springer US.

Holland, J. H. (1992). Genetic algorithms. *Scientific American, 267*(1), 66–72.

Karaboga, D., & Basturk, B. (2007). Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In *International fuzzy systems association world congress* (pp. 789–798). Berlin Heidelberg: Springer.

Koenig, A. C. (2002). *A study of mutation methods for evolutionary algorithms*. University of Missouri-Rolla.

Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*: Vol. 1. MIT press.

Lee, C. Y., & Yao, X. (2001). Evolutionary algorithms with adaptive lévy mutations. In *Evolutionary computation, 2001. Proceedings of the 2001 congress on: Vol. 1* (pp. 568–575). IEEE.

Liang, J. J., Qu, B. Y., Suganthan, P. N., & Hernández-Díaz, A. G. (2013). *Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization*. Singapore: Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University Technical Report, 201212.

Liu, J., & Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. *Soft Computing, 9*(6), 448–462.

Łukasik, S., & Kowalski, P. A. (2015). Study of flower pollination algorithm for continuous optimization. In *Intelligent systems' 2014* (pp. 451–459). Springer International Publishing.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.

Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software, 83*, 80–98.

Nabil, E. (2016). A modified flower pollination algorithm for global optimization. *Expert Systems with Applications, 57*, 192–203.

Ochoa, A., González, S., Margain, L., Padilla, T., Castillo, O., & Melín, P. (2014). Implementing flower multi-objective algorithm for selection of university academic credits. In *Nature and biologically inspired computing (NaBIC), 2014 sixth World congress on* (pp. 7–11). IEEE.

Omran, M. G., Salman, A., & Engelbrecht, A. P. (2005). Self-adaptive differential evolution. In *International conference on computational and information science* (pp. 192–199). Berlin: Springer Heidelberg.

Pambudy, M. M. M., Hadi, S. P., & Ali, H. R. (2014). Flower pollination algorithm for optimal control in multi-machine system with GUPFC. In *Information technology and electrical engineering (ICITEE), 2014 6th International conference on* (pp. 1–6). IEEE.

Pavlyukevich, I. (2007). Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics, 226*(2), 1830–1844.

Prathiba, R., Moses, M. B., & Sakthivel, S. (2014). Flower pollination algorithm applied for different economic load dispatch problems. *International Journal of Engineering and Technology, 6*(2), 1009–1016.

Price, K., Storn, R. M., & Lampinen, J. A. (2006). *Differential evolution: A practical approach to global optimization*. Springer Science & Business Media.

Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation, 13*(2), 398–417.

Ram, J. P., & Rajasekar, N. (2016). A novel flower pollination based global maximum power point method for solar maximum power point tracking. *IEEE Transactions on Power Electronics, PP*(99).

Rechenberg, I. (1978). Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie* (pp. 83–114). Berlin Heidelberg: Springer.

Ronkkonen, J., Kukkonen, S., & Price, K. V. (2005, September). Real-parameter optimization with differential evolution. In *Proc. IEEE CEC: Vol. 1* (pp. 506–513).

Rudolph, G. (1997). Local convergence rates of simple evolutionary algorithms with Cauchy mutations. *IEEE Transactions on Evolutionary Computation, 1*(4), 249–258.

Sakib, N., Kabir, M. W. U., Subbir, M., & Alam, S. (2014). A comparative study of flower pollination algorithm and bat algorithm on continuous optimization problems. *International Journal of Soft Computing and Engineering, 4*(2014), 13–19.

Salgotra, R., & Singh, U. (2016). A novel bat flower pollination algorithm for synthesis of linear antenna arrays. *Neural Computing and Applications*, 1–14.

Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software, 105*, 30–47.

Sharawi, M., Emary, E., Saroit, I. A., & El-Mahdy, H. (2014). Flower pollination optimization algorithm for wireless sensor network lifetime global optimization. *International Journal of Soft Computing and Engineering, 4*(3), 54–59.

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*(6), 702–713.

Singh, U., & Salgotra, R. (2016). Synthesis of linear antenna array using flower pollination algorithm. *Neural Computing and Applications*, 1–11.

Storn, R., & Price, K. (1995). *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*: Vol. 3. Berkeley: ICSI.

Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., & Tiwari, S. (2005). *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization KanGAL report, 2005005*, 2005.

Valenzuela, L., Valdez, F., & Melin, P. (2017). Flower pollination algorithm with fuzzy approach for solving optimization problems. In *Nature-inspired design of hybrid intelligent systems* (pp. 357–369). Springer International Publishing.

Walker, M. (2009). How flowers conquered the world. *BBC Earth News, 10*.

Wang, R., & Zhou, Y. (2014). Flower pollination algorithm with dimension by dimension improvement. *Mathematical Problems in Engineering, 2014*.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82.

Yang, X. S. (2010a). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)* (pp. 65–74). Berlin Heidelberg: Springer.

Yang, X. S. (2010b). Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation, 2*(2), 78–84.

Yang, X. S. (2012). Flower pollination algorithm for global optimization. In *International conference on unconventional computing and natural computation* (pp. 240–249). Berlin Heidelberg: Springer.

Yang, X. S., Karamanoglu, M., & He, X. (2013). Multi-objective flower algorithm for optimization. *Procedia Computer Science, 18*, 861–868.

Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation, 3*(2), 82–102.

Yao, X., Lin, G., & Liu, Y. (1997,). An analysis of evolutionary algorithms based on neighbourhood and step sizes. In *International conference on evolutionary programming* (pp. 297–307). Berlin Heidelberg: Springer.

Zhou, Y., Wang, R., & Luo, Q. (2016). Elite opposition-based flower pollination algorithm. *Neurocomputing, 188*, 294–310.