We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter 13

A Review and Comparative Study of Firefly Algorithm and its Modified Versions

Waqar A. Khan, Nawaf N. Hamadneh,
Surafel L. Tilahun and Jean M. T. Ngnotchouye

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/62472

Abstract

Firefly algorithm is one of the well-known swarm-based algorithms which gained popularity within a short time and has different applications. It is easy to understand and implement. The existing studies show that it is prone to premature convergence and suggest the relaxation of having constant parameters. To boost the performance of the algorithm, different modifications are done by several researchers. In this chapter, we will review these modifications done on the standard firefly algorithm based on parameter modification, modified search strategy and change the solution space to make the search easy using different probability distributions. The modifications are done for continuous as well as non-continuous problems. Different studies including hybridization of firefly algorithm with other algorithms, extended firefly algorithm for multiobjective as well as multilevel optimization problems, for dynamic problems, constraint handling and convergence study will also be briefly reviewed. A simulation-based comparison will also be provided to analyse the performance of the standard as well as the modified versions of the algorithm.

Keywords: Optimization, Metaheuristic Algorithms, Parametric Modification, Mutation, Binary Problems, Simulation

1. Introduction

An optimization problem refers to the maximization or minimization of an objective function by setting suitable values for the variables from a set of feasible values. These problems appear not only in complex scientific studies but also in our day-to-day activities. For instance,

INTECH
open science | open minds

© 2016 The Author(s). Licensee InTech. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. CC BY

when a person wants to go from one place to another and has multiple possible routes, a decision needs to be made on which route to take. The decision can be with the objective to minimize travel time, fuel consumption and so on. However, these kinds of problems with fewer number of alternatives can easily be solved by looking at the outcome of each of the alternatives. However, in real problems, it is not always the case to have a finite and small number of alternatives. Hence, different solution methods are proposed based on the behaviour of the problem.

Since the introduction of evolutionary algorithms, many studies have been conducted on heuristic algorithms. Introducing new algorithms has been one of the leading research areas [1]. Currently, there are more than 40 metaheuristic algorithms [2]. Most of these new algorithms are introduced by mimicking a scenario from nature. For instance, genetic algorithm is inspired by the Darwin theory of survival of the fittest [3]; particle swarm optimization is another metaheuristic algorithm mimicking how a swarm moves by following each other [4]; firefly algorithm is inspired by how fireflies signal each other using the flashing light to attract for mating or to identify predators [5] and prey predator algorithm is another new algorithm inspired by the behaviour of a predator and its prey [6]. These algorithms use different degree of exploration and exploitation based on their different search mechanisms.

Firefly algorithm is among those metaheuristic algorithms which have different applications. Its uncomplicated and easy steps with its effectiveness attract researchers from different disciplines it. Different studies have been performed to modify the standard firefly algorithm to boost its performance and to make it suitable for a problem at hand. In this chapter, a comprehensive study will be presented on firefly algorithm and its modified versions. A brief discussion on extended firefly algorithm with other relevant studies will also be provided. In the next section, a discussion on optimization problems with their solution methods will be given followed by a review on studies on firefly algorithm, which includes a discussion on the standard firefly algorithm with its modified versions and other relevant studies on firefly algorithm, in Section 3. In Section 4, a comparative study based on simulation results will be presented followed by summary of the chapter in Section 5.

2. Optimization problems

Decision-making problems can be found beyond our daily activity. They are very common in engineering, management and in many other disciplines. Different researchers used the concept of optimization in different applications, including engineering applications, transportation planning, management applications, economics, computational intelligence, decision science, agriculture, tourism, sport science and even political science [7–18].

When these problems are formulated mathematically, they are called mathematical optimization problems. It will have a set of feasible actions, also called feasible regions, and a measure of performance of these actions called the objective. A standard single objective minimization problem can be given as in Eq. (1).

http://dx.doi.org/10.5772/62472

$$\min_{x}\{f(x) \mid x \in S \subseteq \Re^n\} \tag{1}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is called the objective function, S is the feasible region and the vector x is the decision variable. A vector $\bar{x}$ is said to be an optimal solution for the minimization problem given in Eq. (1) if and only if $\bar{x} \in S$ and $f(\bar{x}) \leq f(x)$, $\forall\, x \in S$. A local solution x' is a member of S and $f(x') \leq f(x)$, for all x in the neighbourhood of x'.

In a broad sense, optimization solution methods can be categorized as exact and approximate solution methods. Exact solution methods are methods which use an exhaustive search for the exact solution in the solution space. They use mathematical and statistical arguments to get an exact solution. They mainly used calculus-based and iterative procedures. Perhaps Fermat is the first to use a calculus-based argument to solve optimization problems [19]. Iterative methods were first proposed and used by Newton and Gauss [20]. Since then, several exact solution methods are proposed and used in different problems. Branch and bound, simplex method and gradient descent method are good examples of exact solution methods. However, due to complex problems modelled from complex real aspects, it becomes challenging for the deterministic solution methods. This leads to the search of new 'out of the box' way of solving these problems, which in turn gives rise to the birth of metaheuristic solution algorithms.

Metaheuristic algorithms are approximate solution methods for an optimization problem which use a randomness property with an 'educated guess' in their search mechanism and try to improve the quality of the solutions at hand through the iterations, from a randomly generated set of feasible solutions, by exploring and exploiting the solution space. Even though these algorithms do not guaranty optimality, they are tested to give a reasonable and acceptable solution. Furthermore, they have the advantage of not to be affected much by the behaviour of the problem; this makes them useful in many applications. Having a variety of algorithms will give the option to choose a suitable one to solve a problem according to its behaviour.

3. Studies on firefly algorithm

3.1. Introduction

Nature has been an inspiration for the introduction of many metaheuristic algorithms. It has managed to find solution to problems without being told but through experience. Natural selection and survival of the fittest was the main motivation behind the early metaheuristic algorithms. Different animals communicate with each other through different mode of communications. Fireflies use their flashing property to communicate. There are around 2000 firefly species with their own distinct flash patterns. They usually produce a short flash with a certain pattern. The light is produced by a biochemical process called the bioluminescence. The flashing communication is used to attract their mate and also to warn predators. Based on the pattern of the light, a suitable mate will communicate back by either mimicking the same pattern or responding with a specific pattern. It also needs to be noted that the light intensity

decreases through distance; hence, a flashing light emanating from a firefly gets a response from fireflies around it within a visual range of the flash.

In addition to enjoying the beautiful view of a summer sky created by fireflies, they have motivated and have been the centre for many scientific researches [5, 21, 22]. In the sense of optimization, if we consider the fireflies as solution on the landscape of the solution space, then the attraction and movement of fireflies can inspire an optimization algorithm in which solutions follow better (brighter) solutions. Hence, firefly algorithm is motivated and inspired by these properties.

3.1.1. The standard firefly algorithm

Firefly algorithm is a swarm-based metaheuristic algorithm which was introduced by Yang [5]. The algorithm mimics how fireflies interact using their flashing lights. The algorithm assumes that all fireflies are unisex, which means any firefly can be attracted by any other firefly; the attractiveness of a firefly is directly proportional to its brightness which depends on the objective function. A firefly will be attracted to a brighter firefly. Furthermore, the brightness decreases through distance based on inverse square law, as given in Eq. (2).

$$I \prec \frac{1}{r^2} \tag{2}$$

If the light is passing through a medium with a light absorption coefficient γ, then the light intensity at a distance of r from the source can be given as in Eq. (3).

$$I = I_0 e^{-\gamma r^2} \tag{3}$$

where I_0 is light intensity at the source. Similarly, the brightness, β, can be given as in Eq. (4).

$$\beta = \beta_0 e^{-\gamma r^2} \tag{4}$$

A generalized brightness function for $\omega \geq 1$ is given in Eq. (5) [5]. In fact, any monotonically decreasing function can be used.

$$\beta = \beta_0 e^{-\gamma r^{\omega}} \tag{5}$$

In the algorithm, a randomly generated feasible solution, called fireflies, will be assigned with a light intensity based on their performance in the objective function. This intensity will be used to compute the brightness of the firefly, which is directly proportional to its light intensity. For minimization problems, a solution with smallest functional value will be assigned with

http://dx.doi.org/10.5772/62472

highest light intensity. Once the intensity or brightness of the solutions is assigned, each firefly will follow fireflies with better light intensity. For the brightest firefly, it will perform a local search by randomly moving in its neighbourhood. Hence, for two fireflies, if firefly *j* is brighter than firefly *i*, then firefly *i* will move towards firefly *j* using the updating formula given in Eq. (6).

$$x_i := x_i + \underbrace{\beta_0 e^{-\gamma r_{ij}^2}}_{=\beta}(x_j - x_i) + \alpha(\varepsilon() - 0.5) \tag{6}$$

where β_0 is the attractiveness of x_j at $r = 0$, in [5] the author recommended that $\beta_0 = 1$ for implementation, γ is an algorithm parameter which determines the degree in which the updating process depends on the distance between the two fireflies, α is an algorithm parameter for the step length of the random movement and $\varepsilon()$ is a random vector from uniform distribution with values between 0 and 1. For the brightest firefly, x_b, the second expression in Eq. (6) will be omitted, as given in Eq. (7).

$$x_b := x_b + \alpha(\varepsilon() - 0.5) \tag{7}$$

Set algorithm parameters (α, γ)
Set simulation set-up (Number of initial solutions and maximum iteration ($N, MaxGen$))
Randomly generate N initial solutions
for $\text{iteration} = 1 : \text{MaxGen}$
    Compute the brightness, I
    Sort the solution in such a way that , $I_i \geq I_{i-1}, \forall i$
    for $i = 1 : n - 1$
        For $j = i + 1 : n$
            If $I_j > I_i$
                move firefly i towards firefly j
            end if
        end for
    end for
    move firefly N , (x_b), randomly
end for
Report the best solution,

Table 1. The standard firefly algorithm.

These updates of the location of fireflies continue with iteration until a termination criterion is met. The termination criterion can be maximum number of iterations, a tolerance from the

optimum value if it is known or no improvement is achieved in consecutive iterations. The algorithm is summarized in **Table 1**.

3.2. Modified versions of firefly algorithm with critical analysis

Firefly algorithm is efficient and an easy-to-implement algorithm. It is also suitable for parallel implementation. However, researches show that it is slow in convergence and easily gets trapped in local optimum for multimodal problems. In addition, the updates solely depend on current performance and no memory on previous best solutions and performances are kept. That may lead to losing better solutions. Furthermore, since the parameters are fixed, the search behaviour remains to be the same for any condition in all iterations. Hence modifying the standard firefly algorithm to boost its performance has been one of the research issues. Furthermore, the standard firefly algorithm is designed for continuous optimization problems; hence in order to use it for non-continuous problems it needs to be modified and adjusted.

3.2.1. Modification for problems with continuous variables

Basically, there are three classes of modification. Class 1 modification is the modification on the parameters. It is the first category in which the parameters of the algorithm are modified and the same updating mechanisms or formulas are used. Class 2 contains new updating mechanisms. It includes modifications which change part or all of the updating formulas, add mutation operator and the likes. The last category, Class 3, includes modifications on the search space, perhaps with the same updating mechanism it may be easier to switch to another 'easy-to-search' space, and changes in the probability distribution when generating random numbers. The categories are not necessarily disjoint as some of the modifications may fall in multiple classes.

3.2.1.1. Class 1 (parametric modification)

In the standard firefly algorithm, the parameters in Eq. (6) are user-defined constants. Like any other metaheuristic algorithms, the performance of a firefly algorithm depends on these parameter values. They control the degree of exploration and exploitation.

Some of the modifications of firefly algorithm are done by making these parameters variable and adaptive. In recent researches on the modification of firefly algorithms, the parameters α, γ and also r are modified. The modification of α affects the random moment of the firefly, whereas modifying either γ or r affects the degree of attraction between fireflies. Adjusting the brightness at the origin, β_0, has also been done in some researches.

a. Modifying the random movement parameter:-

To deal with parameter identification of infinite impulse response (IIR) and nonlinear systems, firefly algorithm is modified in [23]. The modification with the random movement is based on initial and final step lengths α_0 and α_∞ using $\alpha := \alpha_\infty + (\alpha_0 - \alpha_\infty)e^{-Itr}$. In addition, additional fourth term in the updating process, given by $\alpha\varepsilon(x_i - x_b)$ where x_b is the brightest firefly of all the

http://dx.doi.org/10.5772/62472

fireflies, is added so that firefly algorithm will resemble and have search behaviour like particle swarm optimization. In order to implement this modification, initial and final randomization parameters, α_0 and α_∞, need to be supplied by the user. The randomized parameter is set to decrease exponentially and within a couple of iteration it will vanish. For example, if $\alpha_0 = 1$ and $\alpha_\infty = 0.2$ starting from 0.089 in the first iteration it will decrease to 0.0001 in the seventh iteration. Furthermore, the additional term in the updating formula takes the solution x_i away from x_b, with a given step length $\alpha\varepsilon$. This is in contradiction to the following concept of the current best solution of the algorithm. Assuming that the step length for the new term as well as the randomness is the same, there is only one additional parameter, either α_0 or α_∞ in place of a single parameter α.

Another firefly algorithm with adaptive α is presented in [24]. The modification is given by $\alpha^{(Itr)} := \alpha^{(Itr-1)}\left(\frac{1}{2Itr_{Max}}\right)^{\frac{1}{Itr_{Max}}}$. In addition, two new solutions are generated based on three solutions from the population chosen randomly, the one with the better brightness from the two new solutions, and x_i will replace x_i and pass for the next iteration. It is also used to solve optimal capacitor placement problem [25]. Similar modification of α with additional mutation and crossover operators is also given in [26].

In [27], the randomized parameter is modified based on the number of iterations using $\frac{0.4}{1+e^{0.005(Itr-Itr_{Max})}}$. The simulation results in 16 benchmark problems show that the modification increases the performance of the standard firefly algorithm significantly.

In extending firefly algorithm for multiobjective problems, an adaptive α is proposed and used in [28]. Here α is made adaptive based on the number of iteration and is given by $\alpha := \alpha_0 0.9^{Itr}$. Hence, the step length decreases faster than linearly.

Self-adaptive step firefly algorithm is another modification done to the third term of the updating process by Yu et al. [29]. The step length α is updated based on the historical information and current situation using $\alpha_i^{(Itr+1)} := 1 - \frac{1}{\sqrt{f_b{}^{(Itr)} - (f_i{}^{(Itr)})^2 + (f_i{}^{(Itr)})^2 + 1}}$ where $h_i{}^{Itr} = \frac{1}{\sqrt{(f_{i_{best}}^{(Itr-1)} - f_{i_{best}}^{(Itr-2)})^2 + 1}}$ for $f_b{}^{Itr} = f(x_b)$, $f_i{}^{Itr} = f(x_i)$ after iteration Itr, $f_{i_{best}}^{(Itr-1)}$ and $f_{i_{best}}^{(Itr-2)}$ the best performance of solution x_i until $Itr - 1$ and $Itr - 2$ iterations. Sixteen two-dimensional benchmark problems are used showing that the proposed approach produces better result with smaller standard deviation. It is a promising idea to control the randomized parameter based on the solution's previous and current performances. The update works in such a way that whenever the solution approaches the brightest firefly its step length will decrease, since the performance of the solutions needs to be saved and the memory complexity should be studied.

Another study of modification of the random movement parameter based on the historic performance of the solution is presented in [30]. Based on its best position until current iteration, $x_{i,\text{best}}$, and the global best solution until current iteration, x_{gbest}, $\alpha_i^{(Itr+1)} = \alpha_i^{(Itr)} - (\alpha_i^{(Itr)} - \alpha_{\min})e^{-\frac{Itr\,|x_{gbest} - x_{i,best}|}{MaxGen}}$.

b. Modifying the attraction parameters

The attraction of one firefly by another depends on the light intensity at the source of the brighter firefly as well as on the distance between them and the light absorption coefficient of the medium.

For a small change in the distance between two fireflies results in a quick decrease of the attraction term. To deal with this problem, Lin et al. [31] introduced a virtual distance which will put r in between 0 and 1 and is defined by $r' = \frac{r - r_{\min}}{r_{\max} - r_{\min}}$, where $r_{\min} = 0$ and $r_{\max} = \sqrt{\sum_{i=1}^{d} (x_{\max}(i) - x_{\min}(i))^2}$ for $x_{\max}(i)$ and $x_{\min}(i)$ being the maximum and minimum values of the i^{th} dimension over all fireflies, respectively. Furthermore, β is set as $\beta = \beta_0 \gamma (1 - r')$. In later iterations, the swarm tends to converge around an optimal solution. It means that the distance r decreases and so does $r_{\max}$. However, in most cases, the decreasing rate of r is faster than $r_{\max}$, resulting in a slight increase in β. In order to overcome the possibility of the attraction term dominating the attraction term, the authors proposed a new updating equation in later iterations using $x_i := x_i + \beta(x_j - x_i)\alpha\varepsilon$. Indeed the new updating formula omits the random movement of the firefly. The firefly will only move towards a brighter firefly with a step length of $\beta\alpha\varepsilon$.

Tilahun and Ong [32] suggested that, rather than making $\beta_0 = 1$, it should be a function of the light intensity given by $\beta_0 = e^{I_{0,j} - I_{0,i}}$ for a firefly i to be attracted to j, where $I_{0,j}$ and $I_{0,i}$ are intensity of fireflies j and i at $r = 0$. In addition, moving the brightest firefly randomly may decrease the brightness; hence a direction which improved the brightness will be chosen from m random directions. If such a direction is not among these m directions, it will stay in its current position. The complexity of the algorithm may increase with respect to the new parameter m, and therefore it should be taken into consideration.

Due to the non-repetition and ergodicity of chaos, it can carry out overall searches at higher speeds. Hence, Gandomi et al. [33] proposed a modification on β and γ using chaos functions. This approach has attracted many researchers, and it has been used in different problem domains. The approach is successfully applied using Chebyshev chaos mapping for MRI brain tissue segmentation in [34], for heart disease prediction using Gaussian mapping [35], reliability-redundancy optimization [36] and for solving definite integral problems in [37]. In [38], chaotic mapping is used for β or γ. In addition, α is made to decrease based on the intensity of solutions using $\alpha = \alpha_{\max} - (\alpha_{\max} - \alpha_{\min}) \frac{I_{\max} - I_{mean}}{I_{\max} - I_{\min}}$ where $I_{\max}$, $I_{\text{mean},}$ and $I_{\min}$ are the maximum, the average and the minimum intensities of the solutions.

Another modification in this category is done in [39]. In this chapter, β is modified using $\beta = (\beta_{\max} - \beta_{\min}) e^{-\gamma r^2} + \beta_{\min}$, where $\beta_{\min}$ and $\beta_{\max}$ are user-supplied values. Similar modification is also done in [40].

http://dx.doi.org/10.5772/62472

c. Modifying both the random and attraction movement parameters

To overcome this challenge arising with an increase in the problem dimension and the size of the feasible region, Yan et al. [41] proposed a modification for the standard firefly algorithm. This modification is done on the generalized distance term given in Eq. (5), in which r^{ω} is replaced by $r^{K\sqrt{n(Range)}}$, where K is a constant parameter, n is the dimension of the problem and Range is the maximum range of the dimensions. The parameter α also reduces with iteration from a starting value α_0 to a final value α_{end} linearly. In addition, a firefly is attracted to another firefly if it is brighter and if it is winking. The winking is based on a probability $p_w = 0.5 + 0.1 count_i$, where $count_i$ is the value of a firefly i winking state counter. The larger the counter the greater will be the probability of shifting the state. The maximum counter is five, and after that it will be reset to 0.

In order to solve economic dispatch problem, firefly algorithm is modified in [42]. To increase the exploration property, the authors replaced the Cartesian distance by the minimum variation distance. In addition, they used mutation operator on α but no explanation on how the mutation works is given.

To deal with premature convergence, firefly algorithm has also been modified based on the light intensity [43]. The light intensity difference is defined by $\xi = \frac{\Delta I_{ij}^{(t)}}{\max\{I\} - \min\{I\}}$ for an iteration t. Based on ξ, a modification is done on γ, β and α as follows, $\gamma = \frac{\gamma_0}{r_{\max}^2}$ where $r_{\max} = \max\{d(x_i, x_j) \mid \forall i, j\}$, $\beta_0 = \begin{cases} \xi, & \xi > \eta_1 \\ \eta_1, & \xi \le \eta_1 \end{cases}$ where η_1 is a new parameter and $\alpha = \alpha_0(0.02 r_{\max})$ where $\alpha_0 = \begin{cases} \xi, & \xi > \eta_2 \\ \eta_2, & \xi \le \eta_2 \end{cases}$ for another new algorithm parameter η_2. The modification shows that for two fireflies, the brighter one will have a small attraction and randomness step length compared to the brighter ones.

For the optimal sizing and siting of voltage-controlled distribution generator in distributed network, firefly algorithm is modified and is used in [44]. The problem is to minimize the power loss by selecting optimal location for distributed generations and the power produced. In the modification β_0 = 1, whereas γ and α are modified based on the problem property (location and maximum power per location in each iteration). This modification is done based on the problem characteristic. The effectiveness and quality of a solution for a metaheuristic algorithm depend on the proper tuning of an algorithm parameter as well as on the behaviour of the landscape of the objective function.

Another modification of the standard firefly algorithm to be listed in this category is done in [45]. The randomized parameter α has been made adaptive using $\alpha = \alpha_{\max} - \frac{Itr(\alpha_{\max} - \alpha_{\min})}{MaxGen}$. Furthermore, the distance function has been made to be influenced not by their location in the landscape of the feasible region but the brightness or functional values of the fireflies using $f(x_b) - f(x_i)$. For two fireflies with similar performance in the objective function, they are considered to be near each other.

For path planning of autonomous underwater vehicle, the parameters γ and α of the standard firefly algorithm are modified by $\gamma = \gamma_b + \frac{Itr}{MaxGen}(\gamma_e - \gamma_b)$ for $\gamma_e > \gamma_b$ and $\alpha = \alpha_b + \frac{Itr}{MaxGen}(\alpha_e - \alpha_b)$ for $\alpha_e < \alpha_b$ [46]. Furthermore, the updating formula is defined as $x_i := x_i + \beta(x_i - x_j) + \alpha r\varepsilon$. As the iteration increases, α decreases and γ increases linearly, implying both the randomness movement and the attraction decrease as a function of the iteration. In the updating formula, the random movement is multiplied by the distance.

A similar approach in which the parameters α,β and γ are encoded in the solution is proposed in [47]. Unlike in [44], the update is done using $\psi := \psi + \sigma_\psi N(0, 1)$ where $\sigma_\psi := \sigma_\psi e^{\tau' N(0,1) + \tau N(0,1)}$ for learning parameters τ,τ' and $\psi = \{\alpha, \beta, \gamma\}$. Another modification that can be listed in this category is done in [48]. The parameter γ is modified using $\gamma_{\max} - (\gamma_{\max} - \gamma_{\min})\left(\frac{Itr}{MaxGen}\right)^2$ for $2 \le \gamma_{\max} \le 4$ and $0.5 \le \gamma_{\min} \le 1$. In addition, for a new parameter λ, $\alpha = \alpha_{\max} - (\alpha_{\max} - \alpha_{\min})\left(\frac{Itr - 1}{G_0 - 1}\right)^\lambda$ where G_0 is an iteration number in which $\alpha = \alpha_{\min}$. This results in a decrease in α quicker than linear function if λ is in the range (0, 1), linearly if $\lambda = 1$ and slower than linear function if $\lambda > 1$. Furthermore, in order to overcome the trapping of the solutions in local optimal solution, Gauss distribution is applied to move the brightest solution, x_b, i.e. $x_b := x_b + x_b N(\mu, \sigma)$. This will be applied if the variance of the solutions before a predetermined M iteration is less than a given precision parameter η. The authors also suggested that chaos, particularly cubic mapping, can be used to improve the distribution of the initial solutions.

In [49], γ and α are computed using $\gamma = 0.03 \mid G_1 \mid$ and $\alpha = 0.03 \mid G_2 \mid$ where G_1 and G_2 are generated from Gaussian or normal distribution with mean 0 and variance 1. Supported by two case studies for multivariable proportional–integral–derivative (PID) controller tuning, a similar study was also done in [50]. They used Tinkerbell mapping to tune γ, using $\gamma = \mid G \mid \bar{x} \frac{itr}{MaxGen}$ where $\bar{x}$ has normalized values generated by the Tinkerbell map in the range [0, 1]. In addition to that, α is modified to decrease linearly using $\alpha = (\alpha_{final} - \alpha_{initial}) \frac{Itr}{MaxGen} + \alpha_{initial}$.

3.2.1.2. Class 2 modifications (new updating mechanisms)

The updating mechanism in the standard firefly algorithm is guided by Eqs. (6) and (7). In Class 1 modification, the same updating equations are used but with adaptive preference. Class 2 modifications include modification on the updating equations including modification in the updating process of the best (the brightest) and the worst (the dimmer) solutions changing part of the updating equations and some modification with additional mutation operator.

a. Modifying the movement of the brightest or dimmer firefly

In a high dimensional problem, the exploration is weak which results in premature convergence. To deal with this, two modifications are proposed in [51] for the standard firefly algorithm. That is, for the initial random N solutions, their opposites will be generated, and the best N solutions will be chosen from the N solutions and their opposites where an opposite

http://dx.doi.org/10.5772/62472

number for x is given by $x_{\min} + x_{\max} - x$. The brightest solution x_b will be updated as follows:

$y = x_b$

for $i = 1 : D$ (for all dimensions)

for $j = 1 : N$ (for all the solutions)

$y(i) = x_j(i)$

if [$f(y)$ is better than $f(x_b)$] $x_b = y$ end if

end for

end for

Similar to the previous modification, here also the best solution will improve or will not change in each of the iterations.

Opposition-based learning is also used in [52], to update the dimmer solution x_w, a solution with worst performance, using $x_w = \begin{cases} x_b, & \varepsilon < p \\ x_{\min} + x_{\max} - x_w, & Otherwise' \end{cases}$ for an algorithm parameter p.

Indeed, it relocates the worst solution to a new position that may give the algorithm a good explorative behaviour.

b. Mutation incorporation

Jumper firefly algorithm is a modified firefly algorithm in which a memory on the performance of each of the solution is kept [53]. A criterion called hazard condition is defined, and solutions will be tested based on their previous performance. If they are in hazardous condition, they will be randomly replaced by a new solution. Hence, based on the hazard condition, a mutation can be done by replacing the weak solution based on previous performance by a new solution.

Another modification in this category is done by Kazemzadeh-Parsi [54], where each iteration k random 'newborn' solutions will be generated to replace the weak solutions. In addition to this mutation, highly ranked k_1 solutions from the previous iteration will replace the same number of weak solutions. The other modification is that, rather than having consecutive movement of a firefly towards brighter fireflies, a single combined direction, the average of the directions $\left(\frac{1}{l}\sum_{i=1}^{l} x_j\right)$ for brighter fireflies x_j's, will be computed and used. Similar approach is used in [55]. In the first case, where newborn solutions replace weak solutions, the number of solutions should not be large; otherwise it behaves as an exhaustive search. In the second case, whenever some solutions are replaced by others from the previous solutions, the solutions coming from the previous iteration will more or less perform similar search behaviour as what has been done in the previous iteration.

Another modification of firefly algorithm by introducing new solutions as a mutation or crossover is given in [26, 56]. In addition to adaptive parameter α, they introduced two mutation operators and five crossover operators based on the mutated solutions. The first

mutation operator works by combining randomly selected three solutions, x_{q1}, x_{q2} and x_{q3}, different from x_i, from the solution set $[x_{mute1} = x_{q1} + \varepsilon(x_{q2} - x_{q3})]$ and the second based on the mutated solution from the first mutation operators, the best and worst solutions (for an iteration t $[x_{mute2} = x_{mute1} + \varepsilon t(x_b - x_w)]$. Based on these two solutions, five solutions will be generated, and the best one from the mutated as well as the new five solutions replace x_i. Similar modification of α and two mutation types are also proposed in [24]. In [57], the parameter α is made to adapt using chaotic mapping and mutation operators.

c. New updating strategy

This is another category of Class 2 modification in which the updating formula, given by Eqs. (6) and (7), is modified or changed. The first modification, to mention in this category, is the modification proposed in [57]. For a firefly *i* attracted by another firefly *j*, the search is updated to be in the vicinity of x_j, as given by $x_i := x_j + \beta(x_j - x_i) + \alpha\varepsilon$. Furthermore, after the update, only improving solutions are accepted. Since the update is done based on the location of x_j, the exploration property of points in between the two solutions will not be done, and it will be trapped in local optimum solution provided x_j is a local solution, and the step lengths are small. Through iterations, the solutions will be forced to be in a neighbourhood of the best solution. The diversity of the solutions will also be low.

A similar modification in the vicinity of the brighter firefly is given in [58]. They proposed two updating formulas, with and without division, as the authors name them. The updating formula, without division, is given by $x_i := x_j + \alpha\varepsilon$. Once the fireflies are sorted according to their brightness, increasing with their index, the updating formula will be defined by $x_i := x_j + \frac{\alpha}{j}\varepsilon$, which will decrease α as the brightness increases and which will give a good intensification or exploitation property. In addition, the parameters α and γ are made adaptive. This put this modification in both Class 1 and Class 2 in our classification. Similar discussion holds for a similar work done in [57]. In addition, unlike in [59], there is an attraction term in the direction from the brighter x_j towards x_i, that means moving the brighter firefly in a non-promising direction replaces x_i. Hence, in this sense, the modification in [58] is better as it moves the solution not in a non-promising direction but randomly.

For a data clustering problem, the standard firefly algorithm is modified firefly algorithm [60, 61]. They proposed a new updating formula to increase the influence of the brightest firefly. The new updating formula is given by $x_i := x_i + \beta(x_j - x_i) + \beta_0 e^{-\gamma r_{i,gbest}^2}(x_{gbest} - x_i) + \alpha\varepsilon$. This means that a firefly is not only attracted to brighter fireflies but also by the best solution found so far, x_{gbest} . Suppose there are l brighter fireflies, brighter than x_i, at the end of the iteration the attraction term will be $\sum_{j=1}^{l}\beta(x_j - x_i) + l\beta_0 e^{-\gamma r_{i,gbest}^2}(x_{gbest} - x_i)$. Hence, repeatedly moving a firefly to the best solution increases the attraction step length, and based on the feasible region, it usually may not be acceptable. Furthermore, the global solution found can be an optimal solution, and it may result in the solutions to be forced to follow that local solution rather than exploring other regions in each loop of iteration. In [61], four ten-dimensional benchmark problems and

http://dx.doi.org/10.5772/62472

four twenty-dimensional benchmark problems along with Iris data set are used for clustering. Similar modification is done in [62]. In addition, to update α in a decreasing manner, the updating formula for a solution x_i based on the brighter firefly x_b and the best solution from memory *gbest* with a new algorithm parameter λ is modified as $x_i := x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \beta_0 e^{-\gamma r_{i,best}^2}(x_b - x_i) + \lambda\varepsilon(x_i - gbest) + \alpha\varepsilon$.

Fuzzy firefly algorithm is another modification of the standard firefly algorithm [63]. Even though they start with a wrong claim by saying "*in the standard firefly algorithm, only one firefly in each iteration can affect others and attract its neighbours*", they try to increase the exploration property of the algorithm by adding an additional term in which the top k fireflies attract according to $x_i := x_i + [\beta_0 e^{-\gamma e_{ij}^2}(x_j - x_i) + \sum_{h=1}^{k} A(h)\beta_0 e^{-\gamma e_{ih}^2}(x_h - x_i)]\alpha\varepsilon$, where $A(h) = \frac{f(x_b)}{l(f(x_h) - f(x_b))}$ with l being a new algorithm parameter. The effect of the best k fireflies is doubled, and if this updating mechanism is done for each brighter firefly x_j then its effect is more than double. Furthermore, multiplying the random term with the second expressions affects their step length and deletes the random movement. Hence, it forces the fireflies to follow best k solutions. Exploring other regions is not possible with this update.

Another modification with a new updating formula is proposed in [64] and is given by $x_i := \begin{cases} x_i + \beta(x_j - x_i) + \alpha\varepsilon(x_{\max} - x_{\min}), & \text{if} \varepsilon > 0.5 \\ \frac{\text{MaxGen-Itr}}{\text{MaxGen}}(1-\eta)x_i + \eta x_b \end{cases}$, otherwise where β_0 is computed based on the location of x_i normalized by the locations of fireflies in the search space and γ is computed with a direct relation with β_0 and additional two parameters; η is a value based on the difference between the location of the fireflies.

Diversity-guided firefly algorithm is one of the recent modified versions [65]. The modification is done to make the solutions as diverse as possible with a given threshold. The updating mechanism of the standard firefly is used until diversity of the solution falls beyond the given threshold. The diversity is measured by $\frac{1}{NL}\sum_{i=1}^{N} \| x_i - \bar{x} \|$ where L is the longest diagonal of the feasible region, and $\bar{x}$ is the average position of all fireflies. If the diversity is less than a predefined threshold value, then the updating formula will be $x_i := x_i + \beta(x_j - x_i) + \alpha\varepsilon(x_i - \bar{x})$. The modification proposed is effective in diversifying the solutions by replacing the random movement by moving the solutions away from the average position of fireflies.

In [66, 67], a mutated firefly algorithm is proposed in such a way that the brighter firefly donates some of its features based on a new algorithm parameter called probability of mutation, p_m. The features and their amount copied from the bright firefly are not mentioned. However, based on the context, it seems some components of the vectors for x_i will be replaced by the corresponding components from the brighter firefly x_j. In [66], this mutation operator will replace the updating formula given in Eq. (6) whereas in [67], the mutation will be done after the update is taken place.

In [68], a firefly located at x_i first checks the direction towards other brighter fireflies and looks for the one that improves its performance. If there exists such a solution in which x_i moves

towards the firefly, its brightness increases. Checking the direction towards each of the solution may increase the complexity. Furthermore, in order to escape a local solution some solution should be allowed to decrease its performance; hence in this modification it is highly affected by local optimum solutions especially in misleading problems.

Another modification in this category is introduced to deal with economic dispatch optimization of thermal units [69]. A memory is used to record the best solution found so far. Based on cultured differential evolution, the updating formula is modified as $x_i := x_i + a\beta(gbest - x_i) + b\alpha(\varepsilon() - 0.5)$ where $a = \frac{f(x_i - f(gbest))}{f_{\max} - f_{\min}}$ and $b = x_{\max} - x_{\min}$ for $x_{\max}$ and $x_{\min}$ being the maximum and minimum component of vector x, respectively.

Another modification in this category is presented in [70]. The updating formula becomes $x_i := wx_i + \beta(x_j - x_i) + \alpha\varepsilon()$ for a weighting parameter w given by $w = w_{\max} - (w_{\max} - w_{\min})\frac{Itr}{MaxGen}$.

3.2.1.3. Class 3 modifications (change in search space or probability density functions)

This class of modifications is on the abstract level modification and includes two types of modifications. The first one is changing the solution space to an easy search space, and the second one is on the types of probability distribution that is used to generate a random vector direction for the random movement.

a. Change in search space

In the modified version presented in [71], each component of a solution will be represented by quaternion $x_i(k) = (y_1^{(i)}, y_2^{(i)}, y_3^{(i)}, y_4^{(i)})$ for all components k of x_i, and the updating will be done over the quaternion space. In order to compute the brightness, the Euclidian space is used by changing the quaternion space to the search space by taking the norm function, $x_i(k) = \|(y_1^{(i)}, y_2^{(i)}, y_3^{(i)}, y_4^{(i)})\|$. Even though the search space increases fourfold, it is interesting to zoom in into each component and perform the search for optimal solution. However, since a norm is used to convert quaternion space to the search space, a mechanism to deal with negative values should be studied. A more mathematical support should be provided along with complexity study.

b. Change in probability distribution function

Perhaps the first work which tries to adapt the randomness movement in the updating process is by Farahani et al. [72, 73]. Even though they started with a wrong claim by saying '*In standard Firefly algorithm, firefly movement step length is a fixed value. So all the fireflies move with a fixed length in all iterations*' by ignoring the random variable ε that makes the step length between 0 and α. They updated the step length to decrease with iteration and introduced a new parameter which updates each solution using $x_i := x_i + \alpha\varepsilon(1 - p)$, where p is a random vector from Gaussian distribution. This will increase the randomness property of the algorithm as it randomly moves once using the usual updating equation. The same modification is also employed in [74].

http://dx.doi.org/10.5772/62472

By enhancing the random movement of a firefly algorithm, Levy firefly algorithm is introduced in [75]. This is the first modification made to firefly algorithm with the Levy distribution guiding the random movement by generating a random direction as well as the step length. The update formula is modified as $x_i := x_i + \beta(x_j - x_i) + \alpha\ sign(\varepsilon) \otimes Levy$; $\otimes$ indicates a component-wise multiplication between the random vector from the Levy distribution and the sign vector. Similarly, in [76], Levy distribution is used to guide the random term of the updating formula. In addition, the parameter α is made to decrease with iteration, using $\alpha = \frac{\alpha_{\max}}{Itr^2}$. Furthermore, what they call information exchange between top fireflies will be done. That is, two solutions are randomly chosen from the top fireflies, and a new solution on the line joining the two fireflies near the brightest one will be generated. Similar approach of using Levy distribution with the step length is generated using a chaotic random number and has applications in image enhancement [77]. The same updating using Levy distribution and same formula for α is used in [78]. In addition to these updates, a communication between top fireflies is used in [79]. Levy distribution along with other probability distribution is suggested for the randomized parameter and used in [80, 81].

3.2.2. Modifications for problems with non-continuous variables

Even though firefly algorithm is introduced for continuous problems, due to its effectiveness it has been modified for non-continuous problems as well. In this section, we will look at three classes of modification. The first one is when modifications are made to solve binary problems. The second is for integer-valued problems which include problems whose variable can have discrete values. The last one is mixed problems in which some of the variables are continuous and the rest are non-continuous.

3.2.2.1. Modifications for binary problems

To deal with set covering problem, a binary firefly algorithm is proposed in [82]. There is no modification in the updating process except converting the solution to either one or zero. Three ways of conversion are proposed in [82]. The conversion works dimension wise. After a solution x_i is updated, for each component p of x_i, three rules based on a transfer function T, which will change the new value of x_i in the interval [0, 1], are given with eight transfer functions. The first rule of conversion is given by $x_i(p) = \begin{cases} 1, & \varepsilon < T(x_i(p)) \\ 0, & \text{Otherwise} \end{cases}$. The second is $x_i(p) = \begin{cases} (x_i^{(t)}(p))^{-1}, & \varepsilon < T(x_i^{t+1}(p)) \\ x_i^{(t)}(p), & \text{Otherwise} \end{cases}$, where $(x_i^{(t)}(p))^{-1}$ is the complement of $x_i^{(t)}(p)$ i.e. if $x_i^{(t)}(p)=1$ then $(x_i^{(t)}(p))^{-1}=0$, otherwise $(x_i^{(t)}(p))^{-1}=1$, $x_i^{(t)}(p)$ and $x_i^{(t+1)}(p)$ are the p^{th} components of x_i from the previous iteration and after the update. The last rule is given by $x_i(p) = \begin{cases} (x^*(p)), & \varepsilon < T(x_i^{t+1}(p)) \\ 0, & \text{Otherwise} \end{cases}$ where x^* is the best memory from memory.

Another modification for binary problems which works dimension wise, in each dimension, is presented in [83]. The update formula of the standard firefly algorithm is used. After the

update, the solution will lie in the interval [0, 1] using $\tanh(x_i(p))$ function for the p^{th} component of solution x_i. Based on the user-defined threshold value, $x_i(p)=1$; if the result is greater than the threshold value, then it will be 0. Another similar work of using sigmoid function is given in [84, 85]. In [86–88], tangent hyperbolic sigmoid function is used for discretizing the solution and also in the updating process using $x_i = \begin{cases} x_i + \beta_0 e^{-\gamma r^2}(x_j - x_i) + \alpha\varepsilon, & if\ \varepsilon < |\tanh(\lambda r)| \\ x_i, & else \end{cases}$, where λ is a parameter close to one.

Another discrete firefly algorithm, in order to deal with job, schedule problem is proposed in [89]. Each firefly x_i in the problem will have two classes of index as $x_i(p, q)$ where p represents the jobs and q their priority in that particular firefly. In order to change real values to binary after the update as sigmoid function, $\frac{1}{1+e^{-x_i(p,q)}}$ is used. Based on the values of the sigmoid function for each job p, the one with higher probability in q will be assigned with value 1 and the rest priority for that particular job with 0.

In [90], for a dynamic knapsack problem, firefly algorithm has been modified. The conversion of the solutions is done based on the property of the problem using priority-based encoding. In addition to making the algorithm to suit for the problem, some modifications are done to increase its effectiveness. One of the modifications is that a firefly i moves towards firefly j if j is brighter and $\varepsilon < rank_i^{-\frac{\mathrm{mode}(Itr-1, MaxGen)}{MaxGen}}$, where $rank_i$ is the rank of firefly i in the solution population. If the condition is not met, i.e. if $\varepsilon \geq rank_i^{-\frac{\mathrm{mode}(Itr-1, MaxGen)}{MaxGen}}$, no updating mechanism is mentioned in the chapter. A similar modification is used in [91]. In addition to the discretization done in [90], the authors in [91] proposed adaptive step length given by $\alpha := \left[1 - \varphi e^{-\mathrm{mod}\left(\frac{Itr}{Itr+1}, 1\right)}\right]\alpha$ for a scaling parameter ϕ. Furthermore, after the updates, two additional moves are introduced. The first one is a random flight by 10% of the top fireflies with 0.45 probability. The move will be accepted only if it is improving. The second is a local search of x_b, after 10% of the iterations x_b will do a local search, and the update will be accepted if it is improving. The additional local searches help to improve the quality of the solution. Furthermore, they also mentioned that chaotic mapping can be used to generate random numbers.

Another modification in this category is presented in [92]. In addition to the discretization, they have made α and γ adaptive using $\alpha = \alpha_{\max} - \frac{Itr(\alpha_{\max} - \alpha_{\min})}{MaxGen}$ and $\gamma = \gamma_{\max} e^{\frac{Itr}{MaxGen} \ln \frac{\gamma_{\min}}{\gamma_{\max}}}$. Furthermore, the random movement is replaced by $\alpha L\ (x_b) \mid x_i - x_b \mid$ for a random number $L(x_b)$ from Levy flight centred at x_b. Three discretization methods, the sigmoid, elf function and rounding function are used to change values in the range [0, 1] along with three updating processes. The first one is done on the continuous space, and sigmoid function will be used to change the results to binary; the second one is the update done on the discrete space, and the discretized results can be used and the third one, instead of using the updating formula, uses a probabilistic method based on sigmoid function.

http://dx.doi.org/10.5772/62472

3.2.2.2. Modifications for integer optimization problems

In [93], firefly algorithm has been modified to deal with software modularization as a graph-partitioning problem. Initially, random integer-encoded solutions are generated. The hamming distance, the number of different entries between two solutions with the same index, is used to measure the distance between two solutions. The update is done by switching a number of entries of a firefly by the entries from a brighter firefly.

Another modification in this category is done in [94]. The modification is based on a concept of random key, which is proposed in [95]. The method uses a mapping of a random number space, $[0,1]^D$, to the problem space.

In [96, 97], the standard firefly algorithm is modified for loading pattern enhancement. The generation of random solutions uses random permutation, and the distance between fireflies, $d(x_i, x_j)$, is measured using hamming's distance. The updating process is separated and made sequentially; first the β step, a move due to the attraction, and next the α step, a move due to the random movement. In the β step, first same entries with same index for both fireflies, x_i and x_j, are preserved and then an entry will be copied from x_j if $\varepsilon < \beta$, where $\beta = \frac{1}{1+\gamma d(x_i,\ x_j)^2}$; otherwise the x_i entry will be used. The α step is done using $x_i := Int(x_i + \alpha\varepsilon)$, with a swapping mechanism to preserve feasibility. A similar modification of sequentially applying β step and α step is also used in [98], with additional modification on β and α using $\beta = e^{-\frac{(\max\{P_i\}-p_{ij})^2}{\max\{P_i\}}}$ for $P_{ij} = \varepsilon + \frac{1}{\lceil rank_i - rank_j \rceil}$ and $\alpha = \left\lfloor D - \frac{D}{MaxGen} \right\rfloor$. It is a good idea to adapt and increase the step length with the dimension of the problem. For instance, when $D = 12$, the step length α will start from 11 and decrease to zero in last iterations. However, if the feasible region is in [0, 4], the search in more than 60% of the time α will be at least 4. Hence, the moves in the first 60 plus % are not acceptable because it will force the solution out of the feasible region. Hence, the modification needs to consider the size of the feasible region. Another similar modification with the modifications done in [97] is given in [99] with additional modification to keep the best solution and use it in the updating process. That is based on $\rho = 0.5 + \frac{0.5Itr}{MaxGen}$, and if $\varepsilon > \rho$ the brighter firefly x_j will be replaced by the global best from memory.

For travel salesman problem, firefly algorithm has been modified in [100]. Initial solutions are generated using permutation of D, and each solution is represented as a string of chromosomes of these numbers. The distance between two solutions is computed using $r = \frac{10A}{D}$, where A is the number of different arcs. The movement is done randomly selecting the length of movement between 2 and r and then using inversion mutation towards better solution; if there is no better solution, a random move will be done. Each firefly will produce m solutions and the best N solutions will be chosen to pass to the next generation.

Another modification in this category is proposed in [101]. The decision variables, x_i's, represent assembly sequence. In the update, the random movement is omitted, and the attraction move is done in the discrete space. The attraction direction is computed for each

dimension k using $s_{ji}(k)=\begin{cases}x_j(k), & if\ \ x_j(k)\neq x_i\\ 0, & \text{else}\end{cases}$, and the update will be done by $x_i:=x_i+S_{ji}$ where $S_{ji}=\begin{cases}s_{ji}, & if\quad \alpha\mid\varepsilon-0.5\mid<\beta_0 e^{-\gamma r^2}\\ 0, & else\end{cases}$. In addition, the visual range, dv, which will control a firefly to be influenced by fireflies in tis visual range, is introduced. The visual range is computed by $dv=\begin{cases}\frac{3Itr(dv_{\max}-dv_{\min})}{2(MaxGen-1)}+dv_{\min}, & if\quad Itr<\frac{2}{3}MaxGen\\ dv_{\max}, & \text{otherwise}\end{cases}$ for parameters $dv_{\max}$ and $dv_{\min}$. This means that a firefly will not be attracted to any brighter firefly but to a brighter firefly in a visible range.

Firefly algorithm has been discretized for supply selection problem in [102]. The sum of the absolute differences between the entries is used to measure the distance $r=\sum_{k=1}^{D}\mid x_j(k)-x_i(k)\mid$. In addition, the movement is modified based on the property of the problem using rounding up for step length. In most cases, a modification specific to a problem is effective for that particular problem or a class of problems. However, it is hard to generalize the problems in other domains. It would be interesting to generalize the approach to be tested in other problem domains as well.

3.2.2.3. Modifications for mixed optimization problems

Perhaps the first modification to the standard firefly algorithm in this category is presented in [103]. The updating of solutions is conducted using the updating mechanism of the standard firefly algorithm. To deal with the discrete variables, constraint handling mechanism is used based on penalty function. In addition, the authors proposed two ways to generate a diverse set of random initial solutions. An adaptive random step length is also proposed using similar updating way in [104]. The same approach is improved in [105] by adding a scaling parameter for the random movement based on the difference between the maximum and minimum values for each variable. Portfolio optimization can be expressed as a mean-variance problem which belongs to the group of quadratic mixed-integer programming problems. In [106, 107], firefly algorithm has been extended with the use of rounding function and constraint handling approach. Deb's method [108] is also used for constraint handling. In addition, α is modified using $\alpha:=\alpha\left[1-\left\{1-\left(\frac{10^{-4}}{9}\right)^{\frac{1}{MaxGen}}\right\}\right]$.

3.3. Discussion

Like any metaheuristic algorithm, firefly algorithm is prone to parameter values. It is noticed that changing the parameters based on the search state is effective. Hence, modification on parameters is a direct forward idea to improve the performance of firefly algorithm. As the search proceeds, in order to have a conversion with good precision, the randomness movement must decrease. Hence, the randomness step length, α, is modified to be adaptive in which its value decreases with iteration [23–28, 41, 45, 46, 48]. **Figure 1** shows the graph of the modifications.

http://dx.doi.org/10.5772/62472

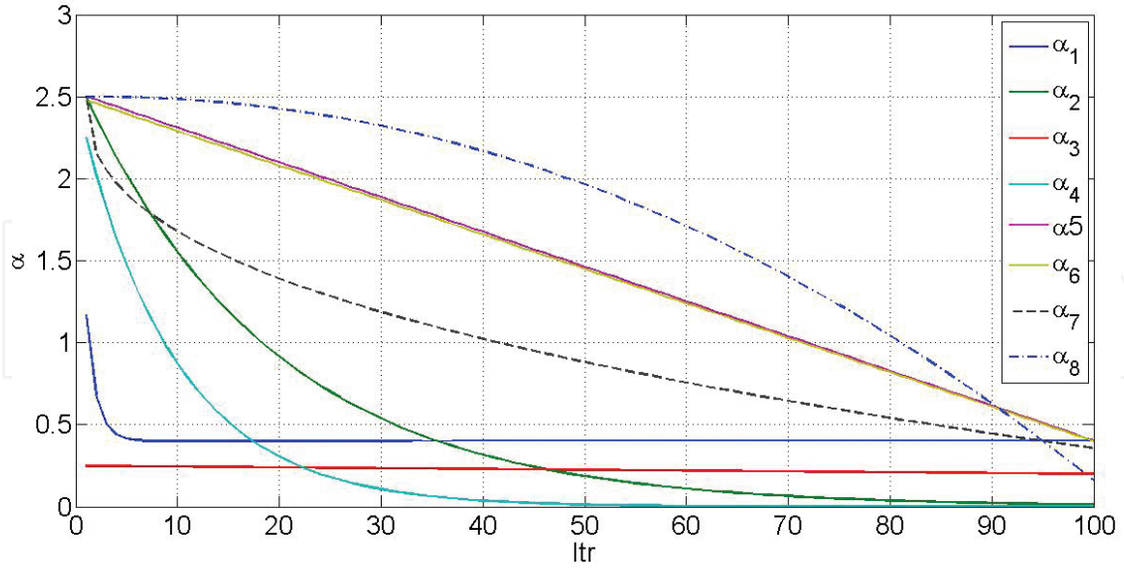


Figure 1. With initial and final values of 2.5 and 0.4; α_1 [23], α_2 [24–26], α_3 [27], α_4 [28], α_5 [41], α_6 [45, 46, 50], α_7 [48] with $\lambda = 0.4$ and α_8 [48] with $\lambda = 2.1$.

The decreasing scenario for α starting from the first iteration may not always be a good idea. Perhaps, it is better to keep a constant α for a number of iterations and start the decreasing scenario based on the performance of the solutions, especially when the solution approaches an optimum point. This can be one of the possible ideas for future work. Some modifications are done making the parameter α adaptive based on the performance of the solution [29, 38, 43]. Some of the modifications also involve a random term, and it behaves neither in decreasing nor in an increasing way [47, 49, 59]. In addition, other approaches such as encoding the parameters in the solution [109] and modifying the parameters based on the problem [44] are also proposed.

The attraction term has also been modified in different ways. Adaptive light absorption constant of the medium changing with iteration is given in some studies. This modifications use increasing function [46], decreasing function [43, 67] or neither increasing nor decreasing function [33–38, 49, 50] of γ. The modification is neither increasing nor decreasing especially when a chaotic distribution [33–38] or normal distribution [34, 49, 50] is used to compute the update. Increasing γ implies the decrease of the attraction step length, and its decrease implies an increase in the step length. The attraction step length β has also been modified. A chaotic mapping is used to modify β in some of the studies [33–38]. It should be noted that using a chaotic map or updating γ does update β. For instance, **Figure 2** shows that when γ is updated using a logistic map, the resulting β is also chaotic.

Hence, γ or β should not be updated at the same time. In addition to γ updating, β has also been done based on minimum and final values [39, 40], depending on the location of the solution [31] and the light intensity of solutions [32]. In addition, different approaches are used to measure the distance between the fireflies [31, 41, 45]. Modifying the feasible region should be considered as a very big step length that may take the solution far away from the brighter solution and possibly out of the feasible region. The property of the random step should also

be properly tuned in agreement with the attraction; otherwise, the random movement may dominate the attractiveness step length.

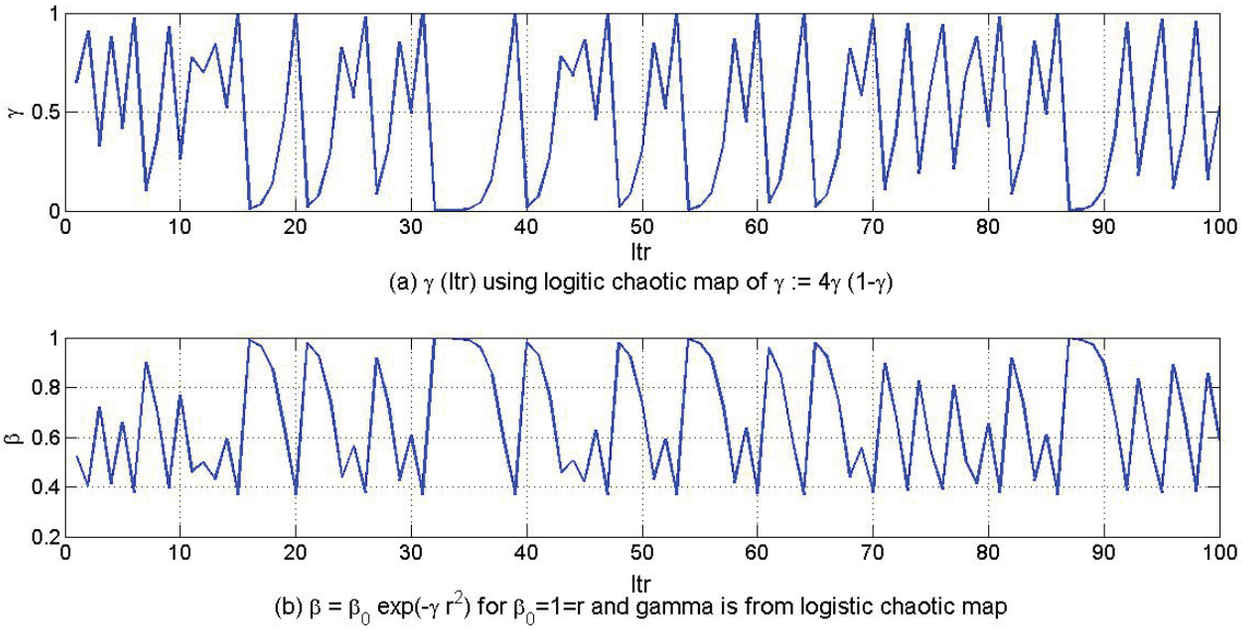


Figure 2. The effect of chaotic map update of γ on β [34].

The movement of the best solution should be tuned properly. If it is allowed to decrease then its best performance may get lost. Hence, the approaches used to preserve the best solution are ended effectively [32, 52].

Mutation is another good approach to diversify the solutions which in turn increase the exploration behaviour of the algorithm [24, 53–58]. However, generating many solutions may hinder the search as it will take long to run. In addition, accepting weak solution should also be incorporated in deceiving problems; a solution needs to decrease in order to escape local solutions.

Modifying the update equation is another interesting modification featured in some studies [56, 58–61, 63–67, 69]. These studies suggest that the update should be done in the vicinity of the brighter firefly [58, 60]. This is not always a good idea as the region in between the two solutions will not be explored. Some of the studies indicate an increase in the attraction towards brighter fireflies [61, 64]. It simply means that increasing the step length of the attraction may dominate the random movement or even take the solution out of the feasible region. A memory is utilized to save the best solution found and additional attraction term towards that global solution is added in [63, 69]. It is a good idea in which the best solution will not be lost through iteration. To increase the diversity of the solution, an effective modification is proposed in [110]. Using such kind of modification, the diversity of the solutions will be preserved, and the exploration behaviour of the algorithm will be improved.

http://dx.doi.org/10.5772/62472

Basically, two updating strategies are proposed for the non-continuous case. The first one is using the same updating formula and changing the results to discrete values afterwards [82, 94, 104]. And the second is to modify the updating formula on the discrete space [97–101]. The first problem is susceptible of trapping in local solution and misses the optimal solution. The optimal solution for a continuous version of a discrete problem may not always be an optimal solution for the discrete problem. Hence, the algorithm will tend to converge to the optimal solution of the continuous version of the problem. Hence, the second approach has an advantage in such cases.

4. Simulation results

The comparison of results is performed between the standard firefly algorithm and non-parameter modified version, i.e. Class 2 and Class 3 modifications. The modified versions selected for simulation are based on two criteria, the first one clear modification, that is the modification should be clearly described, and the second one is with small number of new parameters. In some of the modifications, a number of new algorithm parameters are introduced and tuning this parameter by itself needs another study so they are not included in the simulation. The modified versions used for simulation include Firefly Algorithm 1 [32], FFA2, [52], FFA3 [53], FFA4 [26, 57], FFA5 [24, 59], FFA6 [58] FFA7 [60], FFA8 [61, 62], FFA9 [69], FFA10 [63] where x_i-$g$best is replaced by $g$best-x_i, FFA11 [110], FFA12 [72–74], FFA13 [75–79], FFA14 from [70].

4.1. Benchmark problems and simulation setup

Five benchmark problems are selected from different categories as presented in **Table 2**. The simulations are performed on Intel® Core™i3-3110M CPU @ 2.40 Ghz 64 bit operating system. MATLAB 7.10.0 (R2010a) is used for these simulations. The algorithm parameters are set as given in **Table 2** for dimensions 2 and 5.

4.2. Simulation results and discussion

The simulation results, as presented in **Table 3**, show that some of the algorithms are very expensive in terms of computational time but give a good result, and others have small running time. For instance, in second problem, when the dimension is 2 on average, FFA3 outperforms all with average CPU time of 8.8, whereas FFA1 and FFA2 give a good approximation with smaller average CPU time. In general, it can be seen that FFA4 is very effective but not with the computational time. FFA1 and FFA2 give good approximate results with smaller CPU time compared to FFA4. However, when the dimension increases, FFA2 outperforms FFA1. Perhaps it is due to the fixed random direction m for all the simulations.

Problems	Ref.	Properties of the problem	Parameters and set-up	
			$D = 2$	$D = 5$
f_1 $e^{-\sum_{i=1}^{D} x_i^2} - 2e^{-\sum_{i=1}^{D} (\frac{x_i}{15})^6} \prod_{i=1}^{D} \cos^2 x_i$ $-20 \le x_i \le 20$	[111]	Multimodal Continuous Differentiable Non-separable	$N = 50$ $\alpha = 4$ $\gamma = 2$	$N = 200$ $\alpha = 5$ $\gamma = 2$
$f_2 \sum_{i=1}^{D} \lvert x_i \sin x_i + 0.1 x_i \rvert$ $-10 \le x_i \le 10$	[112]	Multimodal Continuous Non-differentiable Separable Non-scalable	$N = 25$ $\alpha = 3$ $\gamma = 2$	$N = 100$ $\alpha = 4$ $\gamma = 2$
$f_3 \sum_{j=1}^{D} [\sum_{i=1}^{5} p(i) x_j^{5-i}]$ for $p = [0.03779 - 0.84056 - 14.427.134]$ $-1 \le x_i \le 12$	[113]	Multimodal Discontinuous Non-differentiable Separable	$N = 20$ $\alpha = 1.5$ $\gamma = 2$	$N = 100$ $\alpha = 3$ $\gamma = 2$
f_4 $-200e^{0.02\sqrt{\sum_{i=1}^{D} x_i^2}}$ $-32 \le x_i \le 32$	[112]	Unimodal Continuous Differentiable Non-separable Non-Scalable	$N = 60$ $\alpha = 6$ $\gamma = 2$	$N = 250$ $\alpha = 7$ $\gamma = 2$
$f_5 \sum_{i=1}^{D} \varepsilon_i \lvert x_i \rvert^i$ $-5 \le x_i \le 5$	[112]	Unimodal Continuous Non-differentiable Separable Scalable Stochastic	$N = 20$ $\alpha = 1.5$ $\gamma = 2$	$N = 70$ $\alpha = 2.5$ $\gamma = 2$

Table 2. Selected benchmark problems and simulation set-up.

		F1				F2				F3				F4				F5			
	D	2		5		2		5		2		5		2		5		2		5	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
FFA	$f(x^*)$	−0.0195	0.1002	0.00	0.00	0.6696	0.4483	0.0504	0.0135	−6.6745	1.2874	−6.5854	3.6570	−193.75	3.4097	−166.46	7.3072	0.0158	0.0148	0.0477	0.0373
	CPU	1.4	0.3	1.4	0.0	0.3	0.1	0.1	0.0	0.2	0.1	2.7	0.1	1.8	0.4	8.4	2.5	0.2	0.0	0.3	0.1
FFA1	$f(x^*)$	−0.7185	0.4220	0.00	0.00	0.0039	0.0085	0.0109	0.0052	−7.6507	0.00	−17.135	1.8541	−199.61	0.1757	−195.44	1.1286	0.0005	0.0006	0.0002	0.0003
	CPU	1.5	0.3	1.5	0.1	0.4	0.1	0.1	0.0	0.3	0.1	2.8	0.2	1.8	0.4	8.8	2.4	0.3	0.0	0.3	0.0
FFA2	$f(x^*)$	−0.7974	0.4028	0.00	0.00	0.0003	0.0003	0.0000	0.0000	−7.6507	0.00	−19.126	0.0	−199.98	0.0136	−199.98	0.0076	0.0047	0.0092	0.0048	0.0102
	CPU	2.6	0.4	3.5	0.3	1.1	0.1	0.1	0.0	0.4	0.1	4.5	0.4	2.2	0.5	11.3	2.9	0.4	0.1	0.9	0.2
FFA3	$f(x^*)$	−0.0128	0.0903	0.00	0.00	0.4402	0.3768	0.0397	0.0158	−7.2621	0.7406	−8.6751	2.9248	−195.20	2.6004	−177.04	6.5591	0.0085	0.0085	0.0135	0.0125

http://dx.doi.org/10.5772/62472

		F1				F2				F3				F4				F5			
	D	2		5		2		5		2		5		2		5		2		5	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
	CPU	1.3	0.2	1.4	0.1	0.3	0.1	0.1	0.0	0.2	0.1	2.6	0.3	1.6	0.3	8.1	2.2	0.2	0.0	0.2	0.1
FFA4	$f(x^*)$	−1.00	0	−1.00	0.00	0.0	0.0	0.00	0	−7.6507	0.00	−17.977	1.9499	−200	0.0	−200	0.0	0.0	0.0	0.00	0.00
	CPU	44.2	6.7	60.4	3.8	8.8	0.6	1.6	0.2	4.2	0.3	126.5	47	52.56	10.3	574.8	96.4	4	0.4	5	0.8
FFA5	$f(x^*)$	−0.0372	0.1711	0.00	0.00	0.0078	0.0169	0.0113	0.0038	−7.6507	0.00	−17.610	1.2629	−199.58	0.2164	−194.51	1.3993	0.0013	0.0013	0.0009	0.0007
	CPU	37.6	5.8	51.0	2.4	7.8	0.8	1.4	0.1	3.6	0.3	102.5	43.1	45.6	9.3	493.6	80.1	3.4	0.3	4.2	0.8
FFA6	$f(x^*)$	0.0000	0.00	0.00	0.00	0.7534	0.5307	0.0277	0.0091	−7.6507	0.00	−13.323	0.8509	−198.43	0.7722	−193.95	1.4193	0.0051	0.0045	0.0014	0.0008
	CPU	1.4	0.2	1.5	0.1	0.3	0.1	0.1	0.0	0.3	0.1	2.7	0.3	1.6	0.3	8.6	2.5	0.2	0.1	0.3	0.1
FFA7	$f(x^*)$	0.0000	0.00	0.00	0.00	0.0664	0.0330	0.0007	0.0002	−7.6507	0.00	−12.286	1.2151	−199.98	0.0136	−196.05	1.8438	0.0001	0.0002	0.0000	0.0001
	CPU	0.7	0.1	0.8	0.1	0.2	0.1	0.1	0.0	0.2	0.1	1.7	0.2	0.9	0.2	4.0	1.3	0.1	0.0	0.2	0.0
FFA8	$f(x^*)$	0.0000	0.00	0.00	0.00	0.7113	0.5634	0.0499	0.0149	−7.2207	0.8647	−6.2323	3.2674	−193.81	3.7645	−167.58	6.8116	0.0084	0.0085	0.0425	0.0372
	CPU	3.6	0.6	6.8	0.1	1.3	0.1	0.1	0.0	0.9	0.1	7.6	0.7	4.9	0.8	29.4	8.5	0.8	0.1	1	0.1
FFA9	$f(x^*)$	−0.0303	0.15	0.00	0.00	1.1369	0.6702	0.1115	0.0251	−5.4108	1.7004	−1.0048	13.039	−188.39	5.3655	−104.37	12.818	0.0486	0.0399	0.0951	0.0915
	CPU	0.3	0.1	0.2	0.0	0.2	0.1	0.0	0.0	0.2	0.1	0.5	0.1	0.4	0.1	0.7	0.1	0.1	0.0	0.1	0.0
FFA10	$f(x^*)$	0.0000	0.00	0.00	0.00	1.0466	0.6202	0.0403	0.0203	−6.7824	1.0078	−13.245	1.6688	−175.52	12.883	−172.68	10.459	0.0064	0.0055	0.0014	0.0012
	CPU	3.5	0.6	11.1	0.1	1.2	0.1	0.1	0.0	0.9	0.1	9.3	0.9	4.7	0.9	64.2	18.1	1.0	0.1	1.4	0.2
FFA11	$f(x^*)$	0.0000	0.00	0.00	0.00	0.6357	0.4193	0.0514	0.0129	−6.7098	1.0934	−6.1737	3.3915	−194.49	3.3451	−165.94	7.3183	0.0208	0.0186	0.0471	0.0324
	CPU	1.3	0.3	1.4	0.1	0.3	0.1	0.1	0.0	0.3	0.1	2.7	0.3	1.7	0.3	8.3	2.3	0.2	0.1	0.3	0.1
FFA12	$f(x^*)$	0.0000	0.00	0.00	0.00	0.6611	0.5103	0.0493	0.0141	−6.5986	1.3430	−6.8659	4.0703	−193.96	3.4841	−169.34	8.7051	0.0161	0.0160	0.0595	0.0452
	CPU	1.5	0.2	1.5	0.1	0.4	0.1	0.1	0.0	0.3	0.1	2.9	0.3	1.8	0.4	8.4	2.4	0.3	0.0	0.3	0.1
FFA13	$f(x^*)$	0.0000	0.00	0.00	0.00	2.1137	1.4840	0.0977	0.0315	−3.3360	3.2207	−1.1917	12.914	−189.53	4.8277	−148.59	12.7616	0.1356	0.1374	1.4724	2.4610
	CPU	5.4	0.9	16.1	0.2	2.4	0.2	0.3	0.1	1.7	0.1	28	2.5	7.7	1.4	122	30.2	1.7	0.2	1.8	0.2
FFA14	$f(x^*)$	−0.6313	0.29	0.2399	0.4	0.0960	0.0829	0.0107	0.0037	−2.3289	7.0785	−0.4888	11.437	−198.32	0.8034	−193.92	1.4485	0.0046	0.0047	0.0010	0.0006
	CPU	1.5	0.3	1.5	0.1	0.4	0.1	0.1	0.0	0.3	0.1	2.7	0.3	1.8	0.4	8.5	2.5	0.2	0.0	0.3	0.1

Table 3. Simulation results.

5. Conclusion

In this chapter, a detailed review of modified versions of firefly algorithm is presented. The modifications are used to boost its performance for both continuous and non-continuous problems. Three classes of modifications are discussed for continuous problems. The first one being parameter level modification which will improve the performance of the algorithm. The second class is on the updating mechanism level, in which new updating equation or mechanisms are introduced. The last class is in the abstract level in which change of solution space

and probability distribution of the randomness term are discussed. The strength and weakness of the approaches are also presented. Simulation results show that mutation-incorporated firefly algorithm gives better result with larger computational time, whereas versions of firefly algorithm with opposition-based learning and elitist movement for the brighter firefly give approximate solution with smaller computational time. Hence, if a proper way of implementation is used, mutation operator and elitist move of brighter firefly algorithm along with possible implementation of opposition-based approach may perform better.

Author details

Waqar A. Khan[1*], Nawaf N. Hamadneh[2], Surafel L. Tilahun[3] and Jean M. T. Ngnotchouye[3]

*Address all correspondence to: wkhan_2000@yahoo.com

1 Department of Mechanical and Industrial Engineering, College of Engineering, Majmaah University, Majmaah, Saudi Arabia

2 College of Science and Theoretical Studies, Saudi Electronic University, Riyadh, Saudi Arabia

3 School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Pietermaritzburg Campus, South Africa

References

[1] Yang X-S. Review of meta-heuristics and generalised evolutionary walk algorithm. International Journal of Bio-Inspired Computation. 2011;3(2):77–84.

[2] Tilahun SL, ONG HC. Prey-predator algorithm: a new metaheuristic algorithm for optimization problems. School of Mathematical Sciences, [thesis]. Penang Universit Sains Malaysia, Malaysia 2013.

[3] Negnevitsky M. Artificial intelligence: a guide to intelligent systems. New York: Pearson Education Limited, England, 2005.

[4] Kennedy J, Eberhart R. Particle swarm optimization. International Conference on Neural Networks IV; IEEE; Perth, Australia; 1995. 1942–8.

[5] Yang X-S. Nature-inspired metaheuristic algorithms. Luniver Press; UK, 2010.

[6] Tilahun, SL and Ong, HC, International Journal of Information Technology & Decision Making, 14 (6), 1331 – 1352, 2015.

http://dx.doi.org/10.5772/62472

[7] Joshi R. Optimization techniques for transportation problems of three variables. IOSR Journal of Mathematics. 2013;9:46–50.

[8] Tilahun SL, Ong HC. Bus timetabling as a fuzzy multiobjective optimization problem using preference-based genetic algorithm. Promet-Traffic & Transportation. 2012;24(3): 183–91.

[9] Ropponen A, Ritala R, Pistikopoulos EN. Broke management optimization in design of paper production systems. Computer Aided Chemical Engineering. 2010;28:865–70.

[10] Pike J, Bogich T, Elwood S, Finnoff DC, Daszak P. Economic optimization of a global strategy to address the pandemic threat. Proceedings of the National Academy of Sciences. 2014;111(52):18519–23.

[11] Hamadneh N, Tilahun SL, Sathasivam S, Choon OH. Prey-predator algorithm as a new optimization technique using in radial basis function neural networks. Research Journal of Applied Sciences. 2013;8(7):383–7.

[12] Ong HC, Tilahun S. Integration fuzzy preference in genetic algorithm to solve multi-objective optimization problems. The Far East Journal of Mathematical Sciences. 2011;55:165–179.

[13] Tilahun, SL, Kassa, SM and Ong, HC, A New Algorithm for Multilevel Optimization Problems Using Evolutionary Strategy, Inspired by Natural Adaptation, (Ed.: Anthony, P, Ishizuka, M and Lukose, D), PRICAI 2012: Trends in Artificial Intelligence: 12th Pacific Rim International Conference on Artificial Intelligence, Kuching, Malaysia, September 3-7, 2012. Verlag berlin Heidelberg: Springer; 2012, 577–88.

[14] Tilahun SL, Asfaw A. Modeling the expansion of Prosopis juliflora and determining its optimum utilization rate to control the invasion in Afar Regional State of Ethiopia. International Journal of Applied Mathematical Research. 2012;1(4):726–43.

[15] Shimoyama K, Seo K, Nishiwaki T, Jeong S, Obayashi S. Design optimization of a sport shoe sole structure by evolutionary computation and finite element method analysis. Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology. 2011;225(4):179–88.

[16] Hamadneh N, Khan WA, Sathasivam S, Ong HC, Design optimization of pin fin geometry using particle swarm optimization algorithm. PLOS ONE, 8(5): e66080, 2013 DOI 10.1371/journal.pone.0066080.

[17] Hamadneh N, Sathasivam S, Tilahun SL, Choon OH. Learning logic programming in radial basis function network via genetic algorithm. Journal of Applied Sciences (Faisalabad). 2012;12(9):840–7.

[18] Solo AMG. The new interdisciplinary fields of political engineering and computational politics. Political Campaigning in the Information Age. IGI Global, USA, 226–227, 2014.

[19] Tikhomirov VM. Stories about maxima and minima. The American Mathematical Society;Universities Press (India) Pvt. Limited, 1990. Trans. Shenitzer, A, American

Mathematical Socity, Mathematical World Volume: 1; 1991; 187 pp; Softcover MSC: Primary 00; 01; 46; 49; Print ISBN: 978-0-8218-0165-9 Product Code: MAWRLD/1 - See more at: http://bookstore.ams.org/mawrld-1/#sthash.9th31NiP.dpuf USA, 1990.

[20] Yamamoto T. Historical developments in convergence analysis for Newton's and Newton-like methods. Journal of Computational and Applied Mathematics. 2000;124(1):1–23.

[21] Babaoglu O, Binci T, Jelasity M, Montresor A, editors. Firefly-inspired heartbeat synchronization in overlay networks. Cambridge, MA; 2007 SASO'07 1st International Conference on Self-Adaptive and Self-Organizing Systems; Cambridge, MA; 2007 SASO'07 1st International Conference on Self-Adaptive and Self-Organizing Systems, IEEE. DOI: 10.1109/SASO.2007.25, Cambridge, Washington, DC, USA 2007.

[22] Miao Y. Resource scheduling simulation design of firefly algorithm based on chaos optimization in cloud computing. International Journal of Grid and Distributed Computing. 2014;7(6):221–8.

[23] Shafaati M, Mojallali H. Modified firefly optimization for IIR system identification. Journal of Control Engineering and Applied Informatics. 2012;14(4):59–69.

[24] Shakarami MR, Sedaghati R. A new approach for network reconfiguration problem in order to deviation bus voltage minimization with regard to probabilistic load model and DGs. International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering. 2014;8(2):430–5.

[25] Olamaei J, Moradi M, Kaboodi T, editors. A new adaptive modified firefly algorithm to solve optimal capacitor placement problem. 2013 18th Conference on Electrical Power Distribution Networks (EPDC); IEEE; 2013.

[26] Kavousi-Fard A, Samet H, Marzbani F. A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting. Expert Systems with Applications. 2014;41(13):6047–56.

[27] Yu S, Yang S, Su S. Self-adaptive step firefly algorithm. Journal of Applied Mathematics, 2013(2013): 8.

[28] Yang X-S. Multiobjective firefly algorithm for continuous optimization. Engineering with Computers. 2013;29(2):175–84.

[29] Yu S, Yang S, Su S. Self-adaptive step firefly algorithm. Journal of Applied Mathematics, 2013(2013): 8. http://dx.doi.org/10.1155/2013/832718.

[30] Yu S, Su S, Lu Q, Huang L. A novel wise step strategy for firefly algorithm. International Journal of Computer Mathematics. 2014;91(12):2507–13.

[31] Lin X, Zhong Y, Zhang H. An enhanced firefly algorithm for function optimisation problems. International Journal of Modelling, Identification and Control. 2013;18(2): 166–73.

[32] Tilahun SL, Ong HC. Modified firefly algorithm. Journal of Applied Mathematics. 2012(2012): 1–12. DOI:10.1155/2012/467631.

[33] Gandomi A, Yang X-S, Talatahari S, Alavi A. Firefly algorithm with chaos. Communications in Nonlinear Science and Numerical Simulation. 2013;18(1):89–98.

[34] Jansi S, Subashini P. A novel fuzzy clustering based modified firefly algorithm with chaotic map for MRI brain tissue segmentation. MAGNT Research Report. 2015;3(1): 52–8.

[35] Long NC, Meesad P, Unger H. A highly accurate firefly based algorithm for heart disease prediction. Expert Systems with Applications. 2015;42(21):8221–31.

[36] Coelho LDS, de Andrade Bernert DL, Mariani VC, editors. A chaotic firefly algorithm applied to reliability-redundancy optimization. 2011 IEEE Congress on Evolutionary Computation (CEC), IEEE; 2011.

[37] Abdel-Raouf O, Abdel-Baset M, El-henawy I. Chaotic firefly algorithm for solving definite integral. International Journal of Information Technology and Computer Science (IJITCS). 2014;6(6):19.

[38] Khalil A-W. Improved firefly algorithm for unconstrained optimization problems. International Journal of Computer Applications Technology and Research. 2014;4(1): 77–81.

[39] Selvarasu R, Surya Kalavathi M, Rajan A, Christober C. SVC placement for voltage constrained loss minimization using self-adaptive Firefly algorithm. Archives of Electrical Engineering. 2013;62(4):649–61.

[40] Meena S, Chitra K. Modified approach of firefly algorithm for non-minimum phase systems. Indian Journal of Science and Technology. 2015 ;8(23):1–8. DOI: 10.17485/ijst/2015/v8i23/72264

[41] Yan X, Zhu Y, Wu J, Chen H. An improved firefly algorithm with adaptive strategies. Advanced Science Letters. 2012;16(1):249–54.

[42] Sulaiman MH, Daniyal H, Mustafa MW, editors. Modified firefly algorithm in solving economic dispatch problems with practical constraints. 2012 IEEE International Conference on Power and Energy (PECon); IEEE; 2012.

[43] Wang B, Li D-X, Jiang J-P, Liao Y-H. A modified firefly algorithm based on light intensity difference. Journal of Combinatorial Optimization. 2016; 31(3), 1045–1060.

[44] Othman MM, Hegazy YG, Abdelaziz AY. A modified firefly algorithm for optimal sizing and siting of voltage controlled distributed generators in distribution networks. Periodica Polytechnica Electrical Engineering and Computer Science. 2015;59(3):104–9.

[45] Subramanian R, Thanushkodi K. An efficient firefly algorithm to solve economic dispatch problems. International Journal of Soft Computing and Engineering. 2013;2(1):52–5.

[46] Liu C, Zhao Y, Gao F, Liu L. Three-dimensional path planning method for autonomous underwater vehicle based on modified firefly algorithm. Mathematical Problems in Engineering. 2015;2015:1–10. http://dx.doi.org/10.1155/2015/561394

[47] Fister I, Yang X-S, Brest J, Fister Jr I. In: Yang X-S, Cui Z, Xiao R, Gandomi A-H, Karamanoglu M, editors. Memetic self-adaptive firefly algorithm (Ed.: Yang, X-S, Cui, Z, Xiao, R Gandomi, AH and Karamanoglu, M) in Swarm Intelligence and Bio-Inspired Computation: Theory and Applications. 2nd ed. london. Elsevier; :73–102, ISBN: 978-0-12-405163-8, 2013.

[48] Fu Q, Liu Z, Tong N, Wang M, Zhao Y, editors. A novel firefly algorithm based on improved learning mechanism. In International Conference on Logistics Engineering, Management and Computer Science (LEMCS 2015); Atlantis Press; 2015.

[49] dos Santos Coelho L, Mariani VC. Improved firefly algorithm approach applied to chiller loading for energy conservation. Energy and Buildings. 2013;59:273–8.

[50] dos Santos Coelho L, Mariani VC. Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. Computers & Mathematics with Applications. 2012;64(8):2371–82.

[51] Roy AG, Rakshit P, Konar A, Bhattacharya S, Kim E, Nagar AK, editors. Adaptive firefly algorithm for nonholonomic motion planning of car-like system. 2013 IEEE Congress on Evolutionary Computation (CEC), IEEE; 2013.

[52] Verma OP, Aggarwal D, Patodi T. Opposition and dimensional based modified firefly algorithm. Expert Systems with Applications. 2016;44:168–76.

[53] Yu S, Zhu S, Ma Y, Mao D. Enhancing firefly algorithm using generalized opposition-based learning. Computing. 2015 :97(7) 741–754

[54] Kazemzadeh-Parsi M. A modified firefly algorithm for engineering design optimization problems. Iranian Journal of Science and Technology. 2014;38(M2):403–21.

[55] Kazemzadeh-Parsi MJ. Optimal shape design for heat conduction using smoothed fixed grid finite element method and modified firefly algorithm. Iranian Journal of Science and Technology Transactions of Mechanical Engineering. 2015;39(M2):367.

[56] Kazemzadeh-Parsi MJ, Daneshmand F, Ahmadfard MA, Adamowski J. Optimal remediation design of unconfined contaminated aquifers based on the finite element method and modified firefly algorithm. Water Resources Management. 2015;29(8): 2895–912.

[57] Mohammadi S, Mozafari B, Solimani S, Niknam T. An adaptive modified firefly optimisation algorithm based on Hong's Point Estimate Method to optimal operation

management in a micro grid with consideration of uncertainties. Energy. 2013;51:339–48.

[58] Kazemzadeh AS, Kazemzadeh AS. Optimum design of structures using an improved firefly algorithm. International Journal of Optimization and Civil Engineering, 1(2), 327–340, 2011; 2:327–340.

[59] Amiri B, Hossain L, Crawford JW, Wigand RT. Community detection in complex networks: multi–objective enhanced firefly algorithm. Knowledge-Based Systems. 2013;46:1–11.

[60] Amaya I, Cruz J, Correa R. A modified firefly-inspired algorithm for global computational optimization. Dyna. 2014;81(187):85–90.

[61] Hassanzadeh T, Meybodi MR, editors. A new hybrid approach for data clustering using firefly algorithm and K-means. 2012 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP), IEEE; 2012.

[62] Hongwei Z, Liwei T, Dongzheng W. Research on improved firefly optimization algorithm based on cooperative for clustering. International Journal of Smart Home, 9(3), 2015; 9(3):205–214.

[63] Goel S, Panchal V, editors. Performance evaluation of a new modified firefly algorithm. 2014 3rd International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions); IEEE; 2014.

[64] Hassanzadeh T, Kanan HR. Fuzzy FA: a modified firefly algorithm. Applied Artificial Intelligence. 2014;28(1):47–65.

[65] Cheung NJ, Ding X-M, Shen H-B. Adaptive firefly algorithm: Parameter analysis and its application, PloS one. 2014;9(11): 1–12, e112634

[66] Arora S, Singh S, Singh S, Sharma B, editors. Mutated firefly algorithm. 2014 International Conference on Parallel, Distributed and Grid Computing (PDGC); IEEE; 2014.

[67] Arora S, Singh S. Performance research on firefly optimization algorithm with mutation. International Conference on Communication, Computing & Systems (ICCCS–2014); 2014. 168–72.

[68] Banati H, Bajaj M. Fire fly based feature selection approach. International Journal of Computer Science Issues, 8(4): 473–480, 2011.

[69] Maidl G, de Lucena DS, dos Santos Coelho L. Economic dispatch optimization of thermal units based on a modified firefly algorithm, In proceeding of the 2nd International Congress of Mechanical Engineering (COBEM 2013), November 3-7, 2013, Ribeirão Preto, SP, Brazil, 7118–7123, ABCM. 2013.

[70] Tian Y, Gao W, Yan S, editors. An improved inertia weight firefly optimization algorithm and application. 2012 International Conference on Control Engineering and Communication Technology (ICCECT); IEEE; 2012.

[71] Fister I, Yang X-S, Brest J. Modified firefly algorithm using quaternion representation. Expert Systems with Applications. 2013;40(18):7220–30.

[72] Farahani SM, Abshouri A, Nasiri B, Meybodi M. A Gaussian firefly algorithm. International Journal of Machine Learning and Computing. 2011;1(5):448–53.

[73] Farahani S, Abshouri A, Nasiri B, Meybodi M, editors. An improved firefly algorithm with directed movement. Proceedings of 4th IEEE International Conference on Computer Science and Information Technology; Chengdu; 2011, 248–251.

[74] Kanimozhi T, Latha K. An adaptive approach for content based image retrieval using Gaussian firefly algorithm. Emerging intelligent computing technology and applications Volume 375 of the series Communications in Computer and Information Science, 213–218. Springer Berlin Heidelberg; 2013.

[75] Yang, X-S, Firefly algorithm, Levy flights and global optimization, in: Research and Development in Intelligent Systems XXVI (Eds Bramer, M, Ellis, R and Petridis, M), Verlag berlin Heidelberg: Springer London, 209–218 (2010).

[76] Wang G, Guo L, Duan H, Liu L, Wang H. A modified firefly algorithm for UCAV path planning. International Journal of Hybrid Information Technology. 2012;5(3):123–44.

[77] Dhal KG, Quraishi I, Das S. A chaotic Lévy flight approach in bat and firefly algorithm for gray level image enhancement. International Journal of Image, Graphics and Signal Processing (IJIGSP). 2015;7(7):69.

[78] Wang G-G, Guo L, Duan H, Wang H. A new improved firefly algorithm for global numerical optimization. Journal of Computational and Theoretical Nanoscience. 2014;11(2):477–85.

[79] Fateen S-EK, Bonilla-Petriciolet A. Intelligent firefly algorithm for global optimization, In Cuckoo Search and Firefly Algorithm, Volume 516 of the series Studies in Computational Intelligence, 315-330, Verlag berlin Heidelberg: Springer International Publishing Switzerland; 2014.

[80] Hassanzadeh T, Vojodi H, Moghadam AME, editors. A multilevel thresholding approach based on Levy-flight firefly algorithm. 2011 7th Iranian Machine Vision and Image Processing (MVIP), IEEE; 2011.

[81] Sahoo A, Chandra S, editors. Levy-flight firefly algorithm based active contour model for medical image segmentation. 2013 6th International Conference on Contemporary Computing (IC3); IEEE; 2013.

[82] Crawford B, Soto R, Olivares-Suarez M, Palma W, Paredes F, Olguín E, et al. A binary coded firefly algorithm that solves the set covering problem. Science and Technology. 2014;17(3):252–64.

[83] Chandrasekaran K, Simon SP, Padhy NP. Binary real coded firefly algorithm for solving unit commitment problem. Information Sciences. 2013;249:67–84.

[84] Yang Y, Mao Y, Yang P, Jiang Y, editors. The unit commitment problem based on an improved firefly and particle swarm optimization hybrid algorithm. Chinese Automation Congress (CAC), 2013, IEEE; 2013.

[85] Chhikara RR, Singh L, editors. An improved discrete firefly and t-test based algorithm for blind image steganalysis. 2015 6th International Conference on Intelligent Systems, Modelling and Simulation (ISMS); IEEE; 2015.

[86] Farhoodnea M, Mohamed A, Shareef H, Zayandehroodi H. Optimum placement of active power conditioners by a dynamic discrete firefly algorithm to mitigate the negative power quality effects of renewable energy-based generators. International Journal of Electrical Power & Energy Systems. 2014;61:305–17.

[87] Farhoodnea M, Mohamed A, Shareef H, Zayandehroodi H. Optimum placement of active power conditioner in distribution systems using improved discrete firefly algorithm for power quality enhancement. Applied Soft Computing. 2014;23:249–58.

[88] Farhoodnea M, Mohamed A. Optimal placement and sizing of active voltage conditioner in a smart grid using discrete firefly algorithm. Third International Conference on Advances in Engineering Sciences & Applied Mathematics (ICAESAM'2015); London, UK; 2015. 20–5.

[89] Sayadi M, Ramezanian R, Ghaffari-Nasab N. A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. International Journal of Industrial Engineering Computations. 2010;1(1):1–10.

[90] Baykasoğlu A, Ozsoydan FB. An improved firefly algorithm for solving dynamic multidimensional knapsack problems. Expert Systems with Applications. 2014;41(8): 3712–25.

[91] Baykasoğlu A, Ozsoydan FB. Adaptive firefly algorithm with chaos for mechanical design optimization problems. Applied Soft Computing. 2015;36:152–64.

[92] Costa MFP, Rocha AMA, Francisco RB, Fernandes EM. Heuristic-based firefly algorithm for bound constrained nonlinear binary optimization. Advances in Operations Research. 2014 (2014): 1–12. http://dx.doi.org/10.1155/2014/215182

[93] Mamaghani AS, Hajizadeh M, editors. Software modularization using the modified firefly algorithm. 2014 8th Malaysian Software Engineering Conference (MySEC); Langkawi: IEEE; 2014, p. 321 – 324. DOI: 10.1109/MySec.2014.6986037

[94] Oliveira IMSD, Schirru R. A modified firefly algorithm applied to the nuclear reload problem of a pressurized water reactor. International Nuclear Atlantic Conference - INAC 2011; Brazil; 2011.

[95] Bean JC. Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing. 1994;6(2):154–60.

[96] Durkota K. Implementation of a discrete firefly algorithm for the QAP problem within the sage framework. BSc Thesis, Czech Technical University. 2011; 393–403.

[97] Poursalehi N, Zolfaghari A, Minuchehr A. Multi-objective loading pattern enhancement of PWR based on the discrete firefly algorithm. Annals of Nuclear Energy. 2013;57:151–63.

[98] Ishikawa M, Matsushita H. Discrete firefly algorithm using familiarity degree. 2013 Shikoku-Section Joint Convention Record of the Institutes of Electrical and Related Engineers TOKUSHIMA, IEICE Tech. Rep., vol. 113, no. 271, NLP2013-89, 105–108, Oct. 2013.

[99] Poursalehi N, Zolfaghari A, Minuchehr A. A novel optimization method, effective discrete firefly algorithm, for fuel reload design of nuclear reactors. Annals of Nuclear Energy. 2015;81:263–75.

[100] Jati GK, Suyanto, editors. Evolutionary discrete firefly algorithm for travelling salesman problem. Second International Conference, ICAIS 2011; Klagenfurt, Austria; 2011; P. 393–403.

[101] Li M, Zhang Y, Zeng B, Zhou H, Liu J. The modified firefly algorithm considering fireflies' visual range and its application in assembly sequences planning. The International Journal of Advanced Manufacturing Technology. 2016;82(5–8):1381–403.

[102] Kota L. Optimization of the supplier selection problem using discrete firefly algorithm. Advanced Logistic systems. 2012;6(1):117–26.

[103] Gandomi AH, Yang X-S, Alavi AH. Mixed variable structural optimization using firefly algorithm. Computers & Structures. 2011;89(23):2325–36.

[104] Bacanin N, Brajevic I, Tuba M. Firefly algorithm applied to integer programming problems. Proceeding of Recent Advances in Mathematics, 2013; 143–148.

[105] Baghlani A, Makiabadi M, Sarcheshmehpour M. Discrete optimum design of truss structures by an improved firefly algorithm. Advances in Structural Engineering. 2014;17(10):1517–30.

[106] Bacanin N, Tuba M. Firefly algorithm for cardinality constrained mean-variance portfolio optimization problem with entropy diversity constraint. The Scientific World Journal. 2014(2014): 1–16. http://dx.doi.org/10.1155/2014/721521.

[107] Tuba M, Bacanin N, editors. Upgraded firefly algorithm for portfolio optimization problem. 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation (UKSim); Cambridge, IEEE; 2014, 113 – 118.

[108] Deb K. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering. 2000;186(2):311–38.

[109] Selvarasu R, Kalavathi MS. Tcsc placement for loss minimisation using self adaptive firefly algorithm. Journal of Engineering Science and Technology. 2015;10(3):291–306.

[110] Yu S, Su S, Huang L. A simple diversity guided firefly algorithm. Kybernetes. 2015;44(1):43–56.

[111] Gavana A. Global optimization benchmarks and AMPGO [Internet]. Available from: http://infinity77.net/global_optimization/test_functions_nd_X.html#go_benchmark.XinSheYang03 [Accessed: 05/01/2016]

[112] Jamil M, Yang X-S. A literature survey of benchmark functions for global optimization problems. International Journal of Mathematical Modelling and Numerical Optimisation. 4(2), 150–194, 2013.

[113] Tilahun SL, Ong HC, Ngnotchouye JM. Extended prey predator algorithm with a group hunting scenario. Advances in Operations Research. Article Id 7325263, (in press) 2016.

IntechOpen

IntechOpen

IntechOpen