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

4,900
Open access books available

124,000
International authors and editors

140M
Downloads

154
Countries delivered to

TOP 1%
Our authors are among the 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,
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 algo‐
rithm is inspired by the Darwin theory of survival of the fittest [3]; particle swarm optimi‐
zation 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 applica‐
tions. Its uncomplicated and easy steps with its effectiveness attract researchers from differ‐
ent 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 ver‐
sions. 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 simula‐
tion 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, trans‐
portation planning, management applications, economics, computational intelligence, deci‐
sion science, agriculture, tourism, sport science and even political science [7–18].

When these problems are formulated mathematically, they are called mathematical optimi‐
zation 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).
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 \( f(\bar{x}) \leq f(x), \quad \forall \ x \in S \). 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 use 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).

\[ I \propto \frac{1}{r^2} \]  

(2)

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

\[ I = I_0 e^{-\gamma r} \]  

(3)

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

\[ \beta = \beta_0 e^{-\gamma r} \]  

(4)

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

\[ \beta = \beta_0 e^{-\gamma \omega r} \]  

(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
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).

\[
x_i := x_i + \beta_0 e^{-\frac{r}{\beta}} (x_j - x_i) + \alpha(\varepsilon(0) - 0.5)
\]

where \( \beta_0 \) is the attractiveness of \( x_j \) at \( r = 0 \), in [5] the author recommended that \( \beta_0 = 1 \) for implementation, \( \gamma \) is an algorithm parameter which determines the degree in which the updating process depends on the distance between the two fireflies, \( \alpha \) 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).

\[
x_b := x_b + \alpha(\varepsilon(0) - 0.5)
\]

| Set algorithm parameters (\( \alpha, \gamma \) ) |
| Set simulation set-up (Number of initial solutions and maximum iterations \{ N, MaxGen \}) |
| Randomly generate \( N \) initial solutions |
| for iteration = 1 : 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 |
| 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 $\alpha$, $\gamma$ and also $r$ are modified. The modification of $\alpha$ affects the random moment of the firefly, whereas modifying either $\gamma$ or $r$ affects the degree of attraction between fireflies. Adjusting the brightness at the origin, $\beta_0$, 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 $\alpha_0$ and $\alpha_\infty$ using $\alpha = \alpha_\infty + (\alpha_0 - \alpha_\infty)e^{-itr}$. In addition, additional fourth term in the updating process, given by $\alpha \epsilon (x_i - x_b)$ where $x_b$ is the brightest firefly of all the
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, $\alpha_0$ and $\alpha_\infty$, 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 $\alpha_0$ or $\alpha_\infty$ in place of a single parameter $\alpha$.

Another firefly algorithm with adaptive $\alpha$ is presented in [24]. The modification is given by $\alpha^{(Itr)}_{\alpha} = \alpha^{(Itr-1)} \left( \frac{1}{2\varepsilon_{\min}} \right)^{\frac{Itr}{Itr_{\max}}}$. 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 $x_b$ and pass for the next iteration. It is also used to solve optimal capacitor placement problem [25]. Similar modification of $\alpha$ 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 $0.4 + e^{-0.005(Itr - Itr_{\max})}$. 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 $\alpha$ is proposed and used in [28]. Here $\alpha$ 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 $\alpha$ is updated based on the historical information and current situation using $\alpha^{(Itr+1)}_{\alpha} = 1 - \frac{1}{\sqrt{f_{b}^{(Itr-1)} - f_{b}^{(Itr-2)} + (f_{b}^{(Itr-2)} - f_{b}^{(Itr-1)})^2 + 1}}$ where $h_{\varepsilon}^{(Itr)} = \frac{1}{\sqrt{f_{b}^{(Itr-1)} - f_{b}^{(Itr-2)} + (f_{b}^{(Itr-2)} - f_{b}^{(Itr-1)})^2 + 1}}$ for $f_{b}^{(Itr)} = f(x_b)$ after iteration $Itr$, $f_{b}^{(Itr-1)}$ and $f_{b}^{(Itr-2)}$ the best performance of solution $x_b$ until iteration $Itr - 1$ 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, $x_{\text{best}_I}$ and the global best solution until current iteration, $x_{\text{global}_I}$, $\alpha^{(Itr+1)}_{\alpha} = \alpha^{(Itr)} - (\alpha^{(Itr)} - \alpha_{\text{min}}) e^{-\frac{m_{\alpha_{\text{best}}}}{\alpha^{(Itr)} - \alpha_{\text{min}}}}$. 

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

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

287
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

\[
\frac{r - r_{\min}}{r_{\max} - r_{\min}} = \frac{r'}{r_{\max} - r_{\min}},
\]

where \( r_{\min} = 0 \) and \( r_{\max} = \sum_{i=1}^{d} (x_{\max}(i) - x_{\min}(i))^2 \) for \( x_{\max}(i) \) and \( x_{\min}(i) \) being the maximum and minimum values of the \( i^{th} \) dimension over all fireflies, respectively. Furthermore, \( \beta \) 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 \( \beta \). 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

\[
x_i := x_i + \beta (x_j - x_i) \alpha \epsilon.
\]

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 \( \beta \alpha \epsilon \).

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 \( \beta \) and \( \gamma \) 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 \( \beta \) or \( \gamma \). In addition, \( \alpha \) is made to decrease based on the intensity of solutions using

\[
\alpha = \alpha_{\max} - (\alpha_{\max} - \alpha_{\min}) \frac{l_{\max} - l_{\text{mean}}}{l_{\max} - l_{\min}},
\]

where \( l_{\max}, l_{\text{mean}}, \) and \( l_{\min} \) are the maximum, the average and the minimum intensities of the solutions.

Another modification in this category is done in [39]. In this chapter, \( \beta \) is modified using

\[
\beta = (\beta_{\max} - \beta_{\min}) \gamma r^2 + \beta_{\min},
\]

where \( \beta_{\min} \) and \( \beta_{\max} \) are user-supplied values. Similar modification is also done in [40].
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 \( r \) is replaced by \( r_0 K \sqrt{n \text{Range}} \), where \( K \) is a constant parameter, \( n \) is the dimension of the problem and \( \text{Range} \) is the maximum range of the dimensions. The parameter \( r \) also reduces with iteration from a starting value \( r_0 \) to a final value \( r_{\text{end}} \). 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 \( p_w = 0.5 + 0.1 \text{count}_i \), where \( \text{count}_i \) 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 \( \alpha \) 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 \( \xi = \frac{\Delta I(t)}{I_{\text{max}} - I_{\text{min}}} \) for an iteration \( t \). Based on \( \xi \), a modification is done on \( \gamma \), \( \beta \) and \( \alpha \) as follows, \( \gamma = \frac{\eta_1}{r_{\text{max}}} \) where \( r_{\text{max}} = \max\{d(x_i, x_j) \mid \forall i, j\} \), \( \beta_0 = \begin{cases} \xi, & \xi > \eta_1 \\ \eta_1, & \xi \leq \eta_1 \end{cases} \) where \( \eta_1 \) is a new parameter and \( \alpha = \alpha_0(0.02 r_{\text{max}}) \) where \( \alpha_0 = \begin{cases} \xi, & \xi > \eta_2 \\ \eta_2, & \xi \leq \eta_2 \end{cases} \) for another new algorithm parameter \( \eta_2 \). 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 \( \beta_0 = 1 \), whereas \( \gamma \) and \( \alpha \) 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 \( \alpha \) has been made adaptive using \( \alpha = \alpha_{\text{max}} \frac{\text{ln}(\epsilon) - \alpha_{\text{min}}}{\text{MaxGen}} \). 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 \( f(x_i) - f(x_j) \). 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 \( \gamma \) and \( \alpha \) of the standard firefly algorithm are modified by

\[
\gamma = \gamma_b + \frac{I_{tr}}{MaxGen}(\gamma_e - \gamma_b) \text{ for } \gamma_e > \gamma_b \text{ and } \alpha = \alpha_b + \frac{I_{tr}}{MaxGen}(\alpha_e - \alpha_b) \text{ for } \alpha_e < \alpha_b.
\]

Furthermore, the updating formula is defined as

\[
x_i = x_i + \beta(x_i - x_j) + \alpha r \varepsilon.
\]

As the iteration increases, \( \alpha \) decreases and \( \gamma \) 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 \( \alpha, \beta, \gamma \) are encoded in the solution is proposed in [47]. Unlike in [44], the update is done using

\[
\psi = \psi + \sigma \psi N(0,1) \text{ where } \sigma \psi = \sigma \psi e^{\tau N(0,1)+\tau N(0,1)}
\]

for learning parameters \( \tau, \tau' \) and \( \psi = \{\alpha, \beta, \gamma\} \). Another modification that can be listed in this category is done in [48]. The parameter \( \gamma \) is modified using

\[
\gamma_{\text{max}} - (\gamma_{\text{max}} - \gamma_{\text{min}}) \left( \frac{I_{tr}}{MaxGen} \right)^2 \text{ for } 2 \leq \gamma_{\text{max}} \leq 4 \text{ and } 0.5 \leq \gamma_{\text{min}} \leq 1.
\]

In addition, for a new parameter \( \lambda \), \( \alpha = \alpha_{\text{max}} - (\alpha_{\text{max}} - \alpha_{\text{min}}) \left( \frac{I_{tr}}{MaxGen} \right)^n \lambda \)

where \( G_0 \) is an iteration number in which \( \alpha = \alpha_{\text{min}} \). This results in a decrease in \( \alpha \) quicker than linear function if \( \lambda \) 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. \( x_b = x_b + \sigma x_{\text{norm}} \) where \( \sigma x_{\text{norm}} \) has normalized values generated by the Tinkerbell map in the range \([0, 1]\). In addition to that, \( \alpha \) is modified to decrease linearly using

\[
\alpha = (\alpha_{\text{final}} - \alpha_{\text{initial}}) \frac{I_{tr}}{MaxGen} + \alpha_{\text{initial}}.
\]

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

$$y = x_b$$

$$y(i) = y(i)$$

if $f(y)$ is better than $f(x_b)$
end if

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 $x_{w} = \begin{cases} x_b, & \varepsilon < p \\ x_{\text{min}} + x_{\text{max}} - x_{w}, & \text{Otherwise} \end{cases}$ 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 $\frac{1}{l} \sum_{i=1}^{l} r_i$ for brighter fireflies $x_i$’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 $a$, 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 \( \{ x_{\text{mute}} = x_{q1} + \varepsilon(x_{q2} - x_{q3}) \} \) and the second based on the mutated solution from the first mutation operators, the best and worst solutions (for an iteration \( t \)) \( x_{\text{mute}2} = x_{\text{mute}1} + \varepsilon t (x_b - x_w) \). 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 \( \alpha \) and two mutation types are also proposed in [24]. In [57], the parameter \( \alpha \) 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 \( x_i : = x_j + \frac{\alpha \varepsilon}{j} \), which will decrease \( \alpha \) as the brightness increases and which will give a good intensification or exploitation property. In addition, the parameters \( \alpha \) and \( \gamma \) 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 \( x_i : = x_i + \beta (x_i - x_i) + \beta \mu e^{-\gamma r_{gbest}^2 (x_{gbest} - x_i)} + \alpha \varepsilon \). 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 \( \sum_{j=1}^{l} \beta (x_j - x_i) + \beta \mu e^{-\gamma r_{gbest}^2 (x_{gbest} - x_i)} \). 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
being a new algorithm parameter. The effect of the best component of the vectors for
predefined threshold value, then the updating formula will be
modification proposed is effective in diversifying the solutions by replacing the random
length and deletes the random movement. Hence, it forces the fireflies to follow best
threshold. The diversity is measured by
by the corresponding components from the brighter firefly
Furthermore, multiplying the random term with the second expressions affects their step
 updating mechanism is done for each brighter firefly
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
location of \( x_i \) normalized by the locations of fireflies in the search space and \( \gamma \) is computed
with a direct relation with 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 \[ \frac{1}{N L} \sum_{i=1}^{N} \| x_i - \bar{x} \| \] 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
the best solution from
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

\[ x_i = x_i + \beta_0 \exp(-r \gamma_e) (x_j - x_i) + \beta_0 \exp(-r \gamma_r) (x_b - x_i) + \lambda e (x_i - g_{\text{best}}) + \alpha e. \]

Another modification with a new updating formula is proposed in [66] and is given by
\[ x_i = \begin{cases} x_i + \beta (x_j - x_i) + \alpha (x_{\text{max}} - x_{\text{min}}), & \text{if } \epsilon > 0.5 \\ \text{MaxGen} - \frac{\text{Itr}}{\text{MaxGen}} (1 - \eta) x_i + \eta x_b \end{cases}, \]

\[ \text{MaxGen} = \frac{\text{Itr}}{\text{MaxGen}} (1 - \eta) x_i + \eta x_b \]

\[ \beta_0 \text{ is computed based on the location of } x_i \text{ normalized by the locations of fireflies in the search space and } \gamma \text{ is computed with a direct relation with } \beta_0 \text{ and additional two parameters}; \eta \text{ is a value based on the difference between the location of the fireflies.} \]

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 \( x \) 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 \( x \) 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

$$\text{x}_i := \text{x}_i + a\beta \left( g_{best} - \text{x}_i \right) + b\alpha \left( \varepsilon() - 0.5 \right)$$

where $a = f(x_i - f(g_{best}))$ 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

$$\text{x}_i := w\text{x}_i + \beta (\text{x}_j - \text{x}_i) + \alpha \varepsilon()$$

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 $x_i(k) = (y_{1i}(k), y_{2i}(k), y_{3i}(k), y_{4i}(k))$ for all components $k$ of $x_i$, and the updating will be done over the quaternion space. In order to compute the brightness, the Euclidean space is used by changing the quaternion space to the search space by taking the norm function, $x_i(k) = \|y_{1i}(k), y_{2i}(k), y_{3i}(k), y_{4i}(k)\|$. 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 $\varepsilon$ that makes the step length between 0 and $\alpha$. 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].
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

$$x_i = x_i + \beta (x_j - x_i) + a \text{ sign}(\varepsilon) \odot \text{Levy}; \odot \text{ indicates a component-wise multiplication between the random vector from the Levy distribution and the sign vector.}$$

Similarly, in [76], Levy distribution is used to guide the random term of the updating formula. In addition, the parameter $a$ is made to decrease with iteration, using $a = \frac{a_{\text{max}}}{I_{\text{tr}}}$. 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 $a$ 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

$$x_i(p) = \begin{cases} (x_i^{(0)}(p))^{-1}, & \varepsilon < T(x_i^{t+1}(p)) \\ x_i^{(0)}(p), & \text{Otherwise} \end{cases}$$

where $(x_i^{(0)}(p))^{-1}$ is the complement of $x_i^{(0)}(p)$ i.e. if $x_i^{(0)}(p) = 1$ then $(x_i^{(0)}(p))^{-1} = 0$, otherwise $(x_i^{(0)}(p))^{-1} = 1$, $x_i^{(0)}(p)$ 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

\[
x_i = \begin{cases} 
    x_i + \beta \exp(-\gamma r^2 (x_j - x_i)) + \alpha \varepsilon, & \text{if } \varepsilon < |\tanh(\lambda r)| \\
    x_i, & \text{else}
\end{cases}
\]

where \(\lambda\) 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 \(\varepsilon < \text{rank}_i\) if the condition is not met, i.e. if \(\varepsilon \geq \text{rank}_i\), 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

\[
\alpha = \alpha_{\text{max}} - \frac{\lambda r}{\text{MaxGen}} (\alpha_{\text{max}} - \alpha_{\text{min}})
\]

and

\[
\gamma = \gamma_{\text{max}} \exp\left(\frac{\lambda r}{\text{MaxGen}} \ln\left(\frac{\gamma_{\text{max}}}{\gamma_{\text{min}}}\right)\right)
\]

Furthermore, the random movement is replaced by \(\alpha L(x_i) | x_i - x_b|\) for a random number \(L(x_i)\) from Levy flight centred at \(x_i\). 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.
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 $\beta$ step, a move due to the attraction, and next the $\alpha$ step, a move due to the random movement. In the $\beta$ 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 $\alpha$ step is done using $x_i := \text{Int}(x_i + \alpha \varepsilon)$, with a swapping mechanism to preserve feasibility. A similar modification of sequentially applying $\beta$ step and $\alpha$ step is also used in [98], with additional modification on $\beta$ and $\alpha$ using $\beta = e^{-\frac{P_{ij}}{\text{MaxGen}}}$ for $P_{ij} = \varepsilon + \frac{1}{\text{MaxGen}} |\text{rank}_i - \text{rank}_j|$, and $\alpha = D - \frac{D_{\text{MaxGen}}}{\text{MaxGen}}$. 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 $\alpha$ 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 $\alpha$ will be at least 4. Hence, the moves in the first 60% 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 $\rho = 0.5 + \frac{0.5 \cdot \text{Itr}}{\text{MaxGen}}$, 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 $r = \frac{10A}{2r}$, 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 $s_{ji}(k) = \begin{cases} x_j(k), & \text{if } x_j(k) \neq x_i(k) \\ 0, & \text{else} \end{cases}$, and the update will be done by $x_i := x_i + S_{ji}$ where

$$S_{ji} = \begin{cases} s_{ji}, & \text{if } \alpha \left| \epsilon - 0.5 \right| < \beta e^{-\gamma r^2} \\ 0, & \text{else} \end{cases}.$$ 

In addition, the visual range, $dv$, which will control a firefly to be influenced by fireflies in its visual range, is introduced. The visual range is computed by

$$dv = \frac{3\alpha (dv_{\text{max}} - dv_{\text{min}})}{2(\text{MaxGen} - 1)} + dv_{\text{min}} \quad \text{if } \text{Itr} < \frac{2}{\text{MaxGen}}$$

for parameters $dv_{\text{max}}$ and $dv_{\text{min}}$. 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 $r = \sum_{i=1}^{n} |x_j(k) - x_i(k)|$. 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, $\alpha$ is modified using

$$\alpha := \alpha \left[ 1 - \left( \frac{10^{-\gamma}}{9} \right)^{\frac{\text{Itr}}{\text{MaxGen}}} \right].$$

### 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, $\alpha$, 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.
The decreasing scenario for $\alpha$ starting from the first iteration may not always be a good idea. Perhaps, it is better to keep a constant $\alpha$ 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 $\alpha$ 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 $\gamma$. 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 $\gamma$ implies the decrease of the attraction step length, and its decrease implies an increase in the step length. The attraction step length $\beta$ has also been modified. A chaotic mapping is used to modify $\beta$ in some of the studies [33–38]. It should be noted that using a chaotic map or updating $\gamma$ does update $\beta$. For instance, Figure 2 shows that when $\gamma$ is updated using a logistic map, the resulting $\beta$ is also chaotic.

Hence, $\gamma$ or $\beta$ should not be updated at the same time. In addition to $\gamma$ updating, $\beta$ 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 $\gamma$ on $\beta$ [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.
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.
<table>
<thead>
<tr>
<th>Problems</th>
<th>Ref.</th>
<th>Properties of the problem</th>
<th>Parameters and set-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1 = \sum_{i=1}^{D} x_i^2 - 2e^\sum_{i=1}^{D} \prod_{i=1}^{D} \cos^2 x_i$</td>
<td>[111]</td>
<td>Multimodal</td>
<td>$N = 50$ $N = 200$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuous</td>
<td>$\alpha = 4$ $\alpha = 5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differentiable</td>
<td>$\gamma = 2$ $\gamma = 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-separable</td>
<td></td>
</tr>
<tr>
<td>$-20 \leq x_i \leq 20$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_2 = \sum_{i=1}^{D} x_i \sin x_i + 0.1 x_i$</td>
<td>[112]</td>
<td>Multimodal</td>
<td>$N = 25$ $N = 100$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuous</td>
<td>$\alpha = 3$ $\alpha = 4$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-differentiable</td>
<td>$\gamma = 2$ $\gamma = 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Separable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-scalable</td>
<td></td>
</tr>
<tr>
<td>$-10 \leq x_i \leq 10$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_3 = \sum_{i=1}^{D} \sum_{j=1}^{5} p(i)x_j^{3.5}$ for $p = [0.03779 - 0.84056 - 14.427.134]$</td>
<td>[113]</td>
<td>Multimodal</td>
<td>$N = 20$ $N = 100$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuous</td>
<td>$\alpha = 1.5$ $\alpha = 3$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-differentiable</td>
<td>$\gamma = 2$ $\gamma = 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Separable</td>
<td></td>
</tr>
<tr>
<td>$-1 \leq x_i \leq 12$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_4 = \sum_{i=1}^{D} \frac{5}{2} x_i^{2.5}$</td>
<td>[112]</td>
<td>Unimodal</td>
<td>$N = 60$ $N = 250$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuous</td>
<td>$\alpha = 6$ $\alpha = 7$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differentiable</td>
<td>$\gamma = 2$ $\gamma = 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-separable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-scalable</td>
<td></td>
</tr>
<tr>
<td>$-32 \leq x_i \leq 32$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_5 = \sum_{i=1}^{D} \epsilon_i \left</td>
<td>x_i \right</td>
<td>^i$</td>
<td>[112]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuous</td>
<td>$\alpha = 1.5$ $\alpha = 2.5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-differentiable</td>
<td>$\gamma = 2$ $\gamma = 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Separable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scalable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stochastic</td>
<td></td>
</tr>
<tr>
<td>$-5 \leq x_i \leq 5$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Problems</th>
<th>Ref.</th>
<th>Properties of the problem</th>
<th>Parameters and set-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x^*) = -0.0195$ $0.1002$ $0.00$ $0.00$ $0.6696$ $0.4483$ $0.0504$ $0.0135$ $-6.6745$ $1.2874$</td>
<td>FFA</td>
<td>Continuous</td>
<td>$\alpha = 4$ $\alpha = 5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differentiable</td>
<td>$\gamma = 2$ $\gamma = 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-separable</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>1.4</td>
<td>0.3</td>
<td>1.4</td>
</tr>
<tr>
<td>$f(x^*) = -0.7185$ $0.4220$ $0.00$ $0.00$ $0.0039$ $0.0085$ $0.019$ $0.0502$ $-7.6507$ $0.00$</td>
<td>FFA1</td>
<td>Continuous</td>
<td>$\alpha = 4$ $\alpha = 5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differentiable</td>
<td>$\gamma = 2$ $\gamma = 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-separable</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>1.5</td>
<td>0.3</td>
<td>1.5</td>
</tr>
<tr>
<td>$f(x^*) = -0.7974$ $0.4220$ $0.00$ $0.00$ $0.0039$ $0.0035$ $0.0000$ $0.0000$ $-7.6507$ $0.00$</td>
<td>FFA2</td>
<td>Continuous</td>
<td>$\alpha = 4$ $\alpha = 5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differentiable</td>
<td>$\gamma = 2$ $\gamma = 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-separable</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>2.6</td>
<td>0.4</td>
<td>3.5</td>
</tr>
<tr>
<td>$f(x^*) = -0.0128$ $0.0903$ $0.00$ $0.00$ $0.4402$ $0.3768$ $0.0397$ $0.0158$ $-7.2621$ $0.7406$</td>
<td>FFA3</td>
<td>Continuous</td>
<td>$\alpha = 4$ $\alpha = 5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differentiable</td>
<td>$\gamma = 2$ $\gamma = 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-separable</td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>F2</td>
<td>F3</td>
<td>F4</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>1.3</td>
<td>0.2</td>
<td>1.4</td>
</tr>
<tr>
<td><strong>μ</strong></td>
<td>0.1</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.6</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.2</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>-7.6507</td>
<td>0.00</td>
<td>-17.977</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.5</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.6</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.6</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.2</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>-7.6507</td>
<td>0.00</td>
<td>-17.977</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.5</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.6</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.6</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.2</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>-7.6507</td>
<td>0.00</td>
<td>-17.977</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.5</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.6</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.2</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>-7.6507</td>
<td>0.00</td>
<td>-17.977</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.5</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.6</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.2</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>-7.6507</td>
<td>0.00</td>
<td>-17.977</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.5</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.6</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.2</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>-7.6507</td>
<td>0.00</td>
<td>-17.977</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.5</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.6</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.2</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>-7.6507</td>
<td>0.00</td>
<td>-17.977</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.5</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.6</td>
<td>0.3</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>2.2</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>σ</strong></td>
<td>-7.6507</td>
<td>0.00</td>
<td>-17.977</td>
</tr>
</tbody>
</table>

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 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. Khan1*, Nawaf N. Hamadneh2, Surafel L. Tilahun3 and Jean M. T. Ngnotchouye3

*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


[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


