Artif Intell Rev (2019) 51:445–492
https://doi.org/10.1007/s10462-017-9568-0

Continuous versions of firefly algorithm: a review

Surafel Luleseged Tilahun[1] · Jean Medard T. Ngnotchouye[2] · Nawaf N. Hamadneh[3]

Published online: 13 July 2017
© Springer Science+Business Media B.V. 2017

Abstract Firefly algorithm is a swarm based metaheuristic algorithm designed for continuous optimization problems. It works by following better solutions and also with a random search mechanism. It has been successfully used in different problems arising in different disciplines and also modified for discrete problems. Unlike its easiness to understand and to implement; its effectiveness is highly affected by the parameter values. In addition modifying the search mechanism may give better performance. Hence different modified versions are introduced to overcome its limitations and increase its performance. In this paper, the modifications done on firefly algorithm for continuous optimization problems will be reviewed with a critical analysis. A detailed discussion on the modifications with possible future works will also be presented. In addition a comparative study will be conducted using forty benchmark problems with different dimensions based on ten base functions. The result shows that some of the modified versions produce superior results with a tradeoff of high computational time. Hence, this result will help practitioners to decide which modified version to apply based on the computational resource available and the sensitivity of the problem.

Keywords Firefly algorithm · Optimization · Bio-inspired algorithm · Swarm intelligence

1 Introduction

Optimization problems are problems of optimizing a given objective function under a set of constraints. A particular minimization problem can be given as in Eq. (1).

✉ Surafel Luleseged Tilahun
surafelaau@yahoo.com

[1] Department of Mathematical Sciences, University of Zululand, Private Bag X1001, KwaDlangezwa, 3886, South Africa

[2] School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Private Bag X01, Scottsville, 3209, Pietermaritzburg, South Africa

[3] Department of Basic Sciences, College of Science and Theoretical Studies, Saudi Electronic University, Riyadh 11673, Saudi Arabia

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

where x is the decision variable, $f(x)$ is the objective function and S is the feasible region. Since one can switch between a minimization and maximization problem by multiplying the objective function by negative one, we will consider a minimization problem.

The application of optimization solution methods have gone beyond our day to day activity. It has been applied in complex problems arising from different disciplines including, engineering, agriculture, management, economics, food science, politics, music science and the likes (Sweitzer 2008; Tilahun et al. 2012; Grachten et al. 2014; Volpato et al. 2008; Ondrisek 2009; Hamadneh et al. 2012; Hernandez and Fontan 2014; Lucia and Xu 1990; Tilahun and Asfaw 2012; Ropponen et al. 2010; Pike et al. 2014; Tilahun and Ong 2012a, b). Hence, contribution to the solution methods has been done by different professionals in different disciplines. Solution methods for an optimization problem can broadly be categorized as deterministic and non-determinist approaches. Metaheuristic algorithms are among the non-deterministic solution methods. Even though these algorithms do not guarantee optimality they are found to give a reasonable and acceptable solution with appropriate tuning of the algorithm parameters.

Since the introduction of evolutionary algorithms in mid 1970s, many researches have been done on heuristic algorithms. Introducing new algorithms has been one of the leading research areas (Yang 2011). Currently, there are hundreds of these algorithms. 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 (Negnevitsky 2005); Particle swarm optimization mimics how a swarm moves by following each other (Kennedy and Eberhart 1995); Firefly algorithm is inspired by how fireflies signal each other using the flashing light to attract for mating or to identify predators (Yang 2008) and Prey predator algorithm is another metaheuristic algorithm inspired by the interaction between a predator and its prey (Tilahun and Ong 2014; Tilahun et al. 2013). These algorithms use different degree of exploration and exploitation based on their different search mechanisms. In addition to introducing new algorithms merging two or more algorithms to improve the overall performance of the algorithms is also another research area which has been studied extensively. Hybrid algorithms are when multiple algorithms are combined so that the weakness of one algorithm will be compensated by the strength of another. Some of these hybridizations are done in particular to solve a particular problem. Furthermore, the application of metaheuristic algorithms is also another forefront research issue.

Firefly algorithm is a swarm based metaheuristic algorithm inspired by the flashing behavior of fireflies. Randomly generated solutions will be considered as fireflies and each will be assigned with a brightness based on their performance in the objective function. Then a firefly will be attracted towards bright fireflies. The algorithm has became popular due to its easiness to understand as well as to implement. It is also not difficult for parallel implementations. Due to its effectiveness, it has been used in different applications, including engineering applications, decision science applications, computer science applications, economics applications and in medical applications (Pan et al. 2013; Reddy and Sekhar 2014; Tilahun and Ong 2013; Alweshah 2014; Kwiecien and Filipowicz 2012; Poursalehi et al. 2013). In order to increase its effectiveness, different modifications on the standard firefly algorithm have been suggested. Hence, in this paper, a detailed review and analysis of these modifications for continuous optimization problems will be discussed. Review papers have recently been published on firefly algorithm (Fister et al. 2013b; Ali et al. 2014; Ariyaratne and Pemarathne 2015; Khan et al. 2016; Abdelaziz et al. 2015), however, this paper focuses particularly on modification done to improve the performance of firefly algorithm when dealing with continuous problem (interested reader for discrete version of the algorithm can refer to Tilahun and Ngnotchouye 2017) and also unlike the other review papers a detailed analysis on each of the modification with their strength and weakness will be discussed. Furthermore, the literature has expanded quickly and hence very recent modifications are not included in previous review papers. In addition, unlike the previously done research papers, a simulation based comparative study will also be presented.

In the next section a discussion on the standard firefly algorithm will be given followed by a discussion on the modified versions in Sect. 3. In Sect. 4 a discussion on future works along with summarizing the paper will be given. In Sect. 5, a simulation based comparison will be presented followed by a conclusion in Sect. 6.

2 Standard firefly algorithm

Nature has been an inspiration to the introduction of many meta-heuristic algorithms. It has managed to find solutions to problems without being told but through experience. Natural selection and survival of the fittest was the main motivation behind the early metaheuristic algorithms, Evolutionary algorithms. In addition most of metaheuristic algorithms are inspired by a given natural scenario.

Firefly algorithm is a swarm based metaheuristic algorithm which is introduced by Yang (2008). 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; and the attractiveness a firefly is directly proportional to its brightness and depends on the objective function. A firefly will be attracted to a brighter firefly. Furthermore the brightness, or light intensity, decreases through distance based on inverse square law, as given in Eq. (2).

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

If the light is passing through a medium with a light absorbtion 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 equation (4).

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

A generalized brightness function for $m \geq 1$ is given in Eq. (5) Yang (2008). In fact any monotonically decreasing function can be used.

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

In the algorithm randomly generated feasible solutions 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 problem a solution x with smallest functional value will be assigned with highest light intensity. Once the intensity or brightness of the solutions are assigned each firefly will follow fireflies with better light intensity. For the brightest firefly since there is no other brighter firefly to follow it will perform a local search by randomly moving in its neighborhood. Hence, for two fireflies i and j with locations x_i and x_j, respectively, if firefly j is brighter than firefly i, then i will move towards j using the updating formula given in Eq. (6).

Table 1 The standard firefly algorithm

```
Set algorithm parameters (γ, α)
Set simulation set-up (Maximum number of iteration (MaxGen),
                                   Number of initial solutions (N))
Randomly generate N feasible solutions (x_1, x_2, ..., x_N)
for iteration = 1 : MaxGen
    Compute the brightness
    Sort the solutions in such a way that I_i ≥ I_{i−1}, ∀i
    for i = 1 : n − 1
        for j = i + 1 : n
            if (I_i < I_j)
                move firefly i towards firefly j
            end if
        end for
    end for
    move firefly n randomly
end for
```

$$x_i := x_i + \beta_0 e^{-\gamma r^2}(x_j - x_i) + \alpha(\varepsilon() - 0.5) \tag{6}$$

where β_0 is the attractiveness of x_j at $r = 0$ for implementation $\beta_0 = 1$, 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 local search and $\varepsilon()$ is a random vector of appropriate dimension with each component randomly generated from a uniform distribution between zero and one. For the brightest firefly, x_b, the second expression in Eq. (6) will be omitted, as given in Eq. (7).

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

The iteration continues until a termination criterion is met. The termination criterion can be maximum number of iteration, 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 Modified versions of firefly algorithm

The modification of the standard firefly algorithm is done to increase the effectiveness of the algorithm. Basically, there are two types of modifications. The first one is modifications in the parameter level. This is when the algorithm parameters are modified to be adaptive or have a certain structure and the second type is modification on the strategy level. This level includes a modified updating strategy including modified updating formula, added mutation operator and the likes.

3.1 Parameter level modification

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

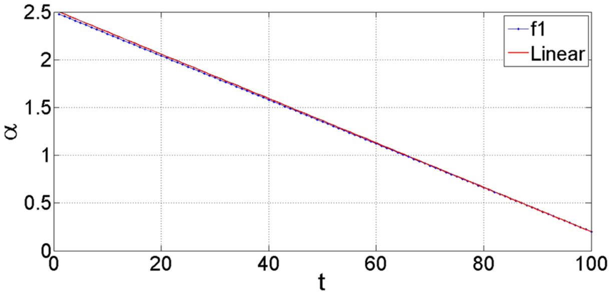


Fig. 1 Adaptive α as in Subramanian and Thanushkodi (2013), Liu et al. (2015), Coelho and Mariani (2012) and in a linear decreasing way

3.1.1 Modifying α

In the standard firefly algorithm, a firefly x_i moves towards better solutions which helps the algorithm to explore around better solutions and also moves randomly. The effect of this random movement depends on the parameter α. If α is chosen to be large then the solution x_i will randomly jump away from the neighborhood and explore the solution space and if it is very small then its jump will be in the neighborhood and also may become negligible compared to the movement towards brighter fireflies. It can also dominate and also move the solution out of the solution space if it is too large. To deal with this different modifications have been proposed. We would also like to point out that, a fixed value of α doesn't mean the movement step length will be α but bounded by α since it is multiplied by a random vector (not-unit vector) whose entries are between -0.5 and 0.5. Some papers purposed a modified α by assuming that the step length is a fixed value but not the upper bound (Farahani et al. 2011a, b).

Some of the modifications done on α is to make it decreasing with iteration. In Subramanian and Thanushkodi (2013), Liu et al. (2015), Coelho and Mariani (2012), a modification is proposed based on preassigned initial and final values for α and using $\alpha = \alpha_{max} - \frac{t(\alpha_{max} - \alpha_{min})}{t_{max}}$, where t is current iteration number, t_{max} is the maximum iteration number, α_{max} is the initial step length and α_{min} is the final step length. This scenario is similar with a linear decreasing scenario, linearly from $(1, \alpha_{max})$ to (t_{max}, α_{min}), proposed in Yan et al. (2012), Goel and Panchal (2014), except that in the linear case it is a slightly larger with a decreasing difference with iteration. If $\alpha_{max} = 2.5$ and $\alpha_{min} = 0.2$ with $t_{max} = 100$, the first values for the linear and the first decreasing case will be 2.5 and 2.477, respectively, and decrease as shown in Fig. 1.

Based on given initial and final step length, an exponential decreasing step length is proposed in Shafaati and Mojallali (2012). The updating formula for α is given by $\alpha = \alpha_{min} + (\alpha_{max} - \alpha_{min})e^{-t}$. It decreases under the linear function. For $\alpha_{max} = 2.5$ and $\alpha_{min} = 0.2$, starting from 1.0641 in the first iteration, at $t = 1$, it will reach to the minimum value under ten iterations. Hence, it may not be suitable where a smooth adapting change is needed, furthermore, it will not be equal to the initial or maximum value even at the beginning, hence when setting up the initial value this needs to be considered and larger value from the intended value should be assigned for α_{max}, perhaps $\alpha_{max} = exp(1)(\alpha_M - \alpha_{min}) + \alpha_{min}$, where α_M is the intended starting step length, can be used.

In addition to modifications of α with a given starting and final step length, there are modification based on an initial step length only. In Wang et al. (2012), α is made to be inversely proportional to the square of the iteration number and given by $\alpha = \frac{\alpha_{max}}{t^2}$. It decreases quickly and the random movement of the firefly will almost vanishes within small

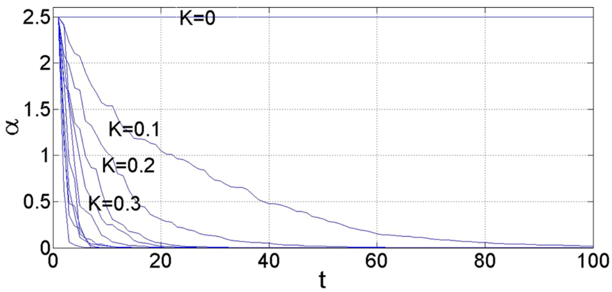


Fig. 2 Adaptive α as given in Amaya et al. (2014) for 100 iteration

number of iterations, for example if $\alpha_{max} = 2.5$, in the second iteration it will be 0.625 and within 16 iterations it will be 0.0098.

In Amaya et al. (2014), $\alpha^{(t)} = \alpha^{(t-1)}(1+K(\varepsilon-1))$, for a random number ε from a uniform distribution between zero and one and K is a new parameter. It should be highlighted that K should be between zero and one, otherwise a negative step length may result. The updating strategy of α decreases quicker than a linear function. Figure 2, shows the step length for $K = 0, 0.1, 0.2 \ldots 0.9, 1$.

Another decreasing scheme for the step length α is proposed in Shakarami and Sedaghati (2014), Olamaei et al. (2013), Kavousi-Fard et al. (2014), using $\alpha^{(t)} = \alpha^{(t-1)}(\frac{1}{2t_{max}})^{\frac{1}{t_{max}}}$, where $\alpha^{(t)}$ and $\alpha^{(t-1)}$ represents the step length in iteration t and $t-1$, respectively. In which the step length starts from α_{max} and it decreases quicker than a linear function and approaches zero when the iteration grows. Unlike the previous two modification, there is no need to give two values as initial and final but a single starting step length. However, it also has a disadvantage of quick decreasing of the step length starting from the beginning which usually may not be needed. A similar modification is applied in Brajevic and Ignjatovic (2015) where $\alpha^{(t)} = \alpha^{(t-1)}(\frac{10^{-4}}{9})^{\frac{1}{t_{max}}}$.

In Yang (2013), Manoharan and Shanmugalakshmi (2015), another decreasing approach is used for α given by $\alpha = \alpha_{max}0.9^t$. It decreases faster than a linear function. If we consider the above example in which $\alpha_{max} = 2.5$, it will start with 2.25 in the first iteration, hence, the maximum step length needs to be put a little higher than the intended starting step length. Like the previous case providing an initial step length is enough, no need to give a final value for α. A generalized form of this modification which is $\alpha = \alpha_{max}\theta^t$ for $\theta \epsilon (0, 1]$ is given in Baghlani et al. (2013). Here θ is new algorithm parameter to control the step length. Figure 3, shows the behaviour of the adaptive α for $\theta = 0.1, 0.2, \ldots, 0.9$ with $\alpha_{max} = 2.5$. Note that if $\theta = 1$, the $\alpha = \alpha_{max}$ for all the iterations. From the figure it can be seen that when θ decreases α also decreases quicker.

Another similar modification with additional parameter is proposed in Fu et al. (2015). α is updated using $\alpha = \alpha_{max} - (\alpha_{max} - \alpha_{min})(\frac{t-1}{G_0-1})^{\lambda}$. Even though the authors did not mention λ should be non-negative, otherwise negative step length may result. If $\lambda = 0$, then for any iteration $\alpha = \alpha_{min}$ and if $\lambda = 1$ then α will decrease linearly. For $\lambda < 1$ α will decrease quicker than a linear function and if $\lambda > 1$ it will decrease slower than a linear function as demonstrated in Fig. 4, for $\alpha_{max} = 2.5$, $\alpha_{min} = 0.2$ and $G_0 = 95$ for $t_{max} = 100$. As can be seen from Fig. 4, if G_0 is not properly chosen then the values of α may go under zero in final iterations.

In Yu et al. (2015a), a modification for the random step length is given by $\alpha = \frac{0.4}{1+e^{0.005(t-t_{max})}}$. α decreases almost in a linear way from 0.2489 in the first iteration and

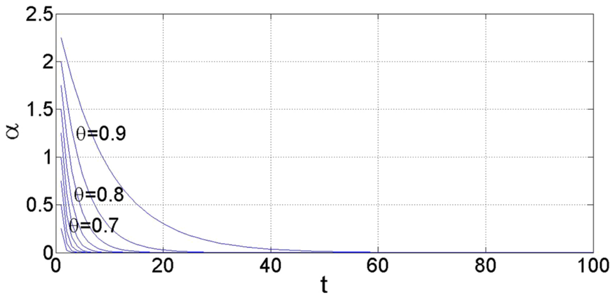
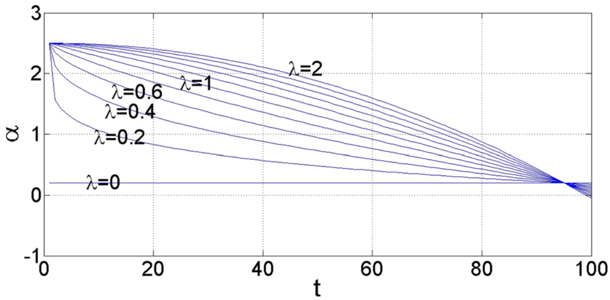


Fig. 3 Adaptive α as in Baghlani et al. (2013) with different values of the new algorithm parameter θ

Fig. 4 Adaptive α as in Fu et al. (2015) with different values of the new algorithm parameter λ



0.2 in the final 100 iteration. Even though it is a decreasing function, α will be too small, unless accompanied by a scaling parameter, for problems with wide feasible region. Another similar modification is given by assigning the inverse of golden ratio as a step length α, is presented in Dhal et al. (2015b).

In Othman et al. (2015), the algorithm parameters are modified based on the problem properties and characteristics. α is made to be inversely proportional to the square of the iteration number, which results a quick decrease as the iteration increases.

In Wang et al. (2014), the authors mentioned that the step length deceases but no decreasing equation is provided.

The summary of the decreasing scenarios of the modifications, $\alpha_{max} = 2.5$ and $\alpha_{min} = 0.2$, is summarized in Fig. 5.

A modification of α based on the performance of the solutions is proposed in AL-Wagih (2015), Wang et al. (2014b), Yu et al. (2013). In AL-Wagih (2015), α is updated using $\alpha = \alpha_{max} - (\alpha_{max} - \alpha_{min})\frac{I_{max}-I_{mean}}{I_{max}-I_{min}}$, where I_{mean} is the average intensity of the fireflies in an iteration. This is similar with the updates done in Subramanian and Thanushkodi (2013), Liu et al. (2015), Coelho and Mariani (2012) with the only difference being rather than the iteration ratio here intensity is used. $\frac{I_{max}-I_{mean}}{I_{max}-I_{min}}$ is always non-negative number at most equal to one. However, it should be highlighted that this updating formula works as far as all the solutions doesn't converge to a global or local solution, because if that is the case a singularity case will be arise with $I_{max} - I_{min} = 0$. Considering, singularity is not the case, the denominator always is greater or equal to the numerator, hence α is always in the range between α_{max} and α_{min}, but not necessarily decreasing.

In Wang et al. (2014b) also an update of α based on the intensity of the population is proposed. In order to compute the step length first a value, ξ, based on change in light intensity of x_i and brighter firefly x_j is calculated using $\xi = \frac{I_j - I_i}{\max(I)-\min(I)}$. Then $\alpha_0 =$

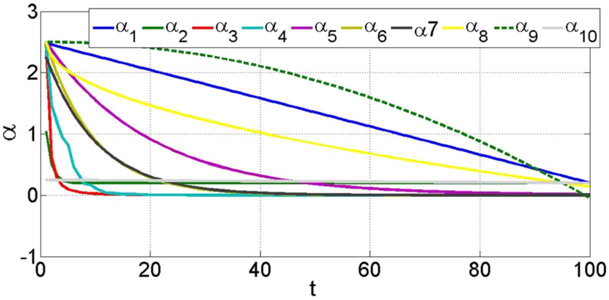


Fig. 5 Adaptive α based on different modifications, α_1 is from Subramanian and Thanushkodi (2013), Liu et al. (2015), Coelho and Mariani (2012); α_2 is from Shafaati and Mojallali (2012); α_3 is from Wang et al. (2012); α_4 is from Amaya et al. (2014) with $K = 0.5$; α_5 is from Shakarami and Sedaghati (2014), Olamaei et al. (2013), Kavousi-Fard et al. (2014); α_6 is from Brajevic and Ignjatovic (2015); α_7 is from Yang (2013), Manoharan and Shanmugalakshmi (2015), Baghlani et al. (2013) with $\theta = 0.9$ in Baghlani et al. (2013); α_8 is from Fu et al. (2015) with $\lambda = 2$; α_9 is from Fu et al. (2015) with $\lambda = 0.5$ and α_{10} is from Yu et al. (2015a) for 100 iteration

$\{\begin{matrix} \xi & \xi > \eta \\ \eta & \xi \leq \eta \end{matrix}$ for a new algorithm parameter η using this α is updated by $\alpha = \alpha_0(0.02r_{max})$ where $r_{max} = \max\{d(x_i, x_j), \forall i, j\}$. For smaller value of η, if the light intensity difference between two fireflies is bigger, then it will result a bigger step length. That is an acceptable relation between the step length and the performance of solutions. However, if η is set bigger, the step length will remain to be the same for any change in the brightness.

In Yu et al. (2013), α is made adaptive based on the performance of previous iterations of the solutions. It utilizes memory to save the performance of the solutions. The updating formula for α is given by $\alpha = 1 - \frac{1}{\sqrt{(f_{best}-f_i)^2+h_i^2 1}}$ for $h_i = \frac{1}{\sqrt{(f_{i-best}^{(t-1)}-f_{i-best}^{(t-2)})^2+1}}$, where f_{best} is the functional value of the best solution solution so far, $f_i = f(x_i)$, $f_{i-best}^{(t-1)}$ is the best functional value of x_i until previous iteration, $t = t - 1$ and $f_{i-best}^{(t-2)}$ the best functional value of x_i until two iteration earlier. It is a good idea to adjust the step length based on the performance of the algorithm. However, As can be seen α is always between zero and one, hence unless a scaling factor is added it may be too small for problems with huge feasible region. Whenever the solution x_i approaches the best solution α will be decreasing. Hence, it is a promising modification based on the performance of the solution, rather than taking a simple decreasing function.

The random step length is not always modified as in a decreasing way. For instance in Coelho and Mariani (2013), it is modified using $\alpha = 0.3|G|$, where G is from a normal distribution of mean 0 and variance 1. Figure 6 shows the step length as a function of iteration for 100 iterations. A scaling parameter should be incorporated which adjusts the length based on the the size of the feasible regions as done for example in Maidl et al. (2013), where α is multiplies by a scaling parameter which is given by $\max_k x_i(k) - \min_k x_i(k)$. In a similar way chaos mapping and other distributions have been used in Amiri et al. (2013), Coelho et al. (2011), Fister et al. (2014), which will produce a non-decreasing or non-increasing step length. A chaotic mapping with Levy flight is also used in Dhal et al. (2015a) and in Yang (2010), Sahoo and Chandra (2013) a Levy distribution is used to generate the random vector for the random movement. Scaling parameter for the random movement is also utilized in Farahani et al. (2011a, b), Liu et al. (2015), Brajevic and Ignjatovic (2015), Baghlani et al. (2013), Kanimozhi and Latha (2013).

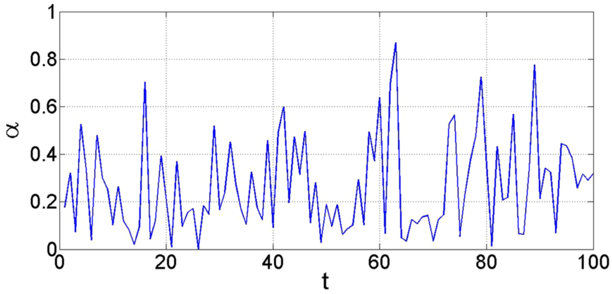


Fig. 6 Adaptive α as given in Coelho and Mariani (2013) for 100 iteration

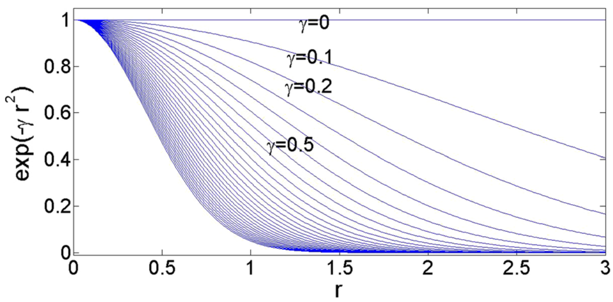


Fig. 7 $\beta = e^{-\gamma r^2}$ as a function of r for $0 \leq r \leq 3$ and $0 \leq \gamma \leq 3$

Another modification done to α is in Selvarasu and Kalavathi (2015). α is encoded as a parameter in the solution along with other parameters. So that the algorithm adjusts the parameter values itself. It is an interesting idea, however, the complexity issue needs to be studied further. In Sulaiman et al. (2012), the authors mentioned that a mutation operator is used to tune α but no description on the operator is given.

3.1.2 Modifying attraction step length, β

The updating formula of a firefly algorithm has two updating terms as given in Eq. (6). The second term, which is given by $\beta_0 e^{-\gamma r^2}(x_j - x_i) = \beta(x_j - x_i)$, represents an attraction term of x_i towards x_j with β_0 has a value based on the light intensity of firefly j at $r = 0$. In the standard firefly algorithm it is suggested that $\beta_0 = 1$. The value of β gives the step length of x_i towards x_j. If $\beta = 1$ then updating x_i towards x_j will put x_i in $x_j's$ position. The attraction step length, β, depends on the initial attraction β_0 based on the light intensity at the source, the distance between the fireflies, r, and the light absorbtion constant, γ. If this step length $\beta = 0$ then x_i will not be attracted and hence will not move towards x_j. If $0 < \beta < 1$ then x_i moves towards x_j and updated to a new position on the line joining the two fireflies. However if $\beta > 1$, x_i will be updated and moved beyond x_j in the direction from x_i to x_j. Hence, if β is assigned with a value in the neighborhood of 1, x_i will move in the neighborhood of x_j and the intensification or exploitation degree of the algorithm around x_j is done. For $\beta_0 = 1$, as suggested by the standard firefly algorithm, and with $0 \leq \gamma \leq 3$ and $0 \leq r \leq 3$, $\beta = e^{-\gamma r^2}$ the scenarios are presented in Fig. 7.

In recent studies modifying the attraction step length is mainly done by modifying γ and β_0. In Selvarasu et al. (2013), based on user assigned β_{max} and β_{min}, β is represented

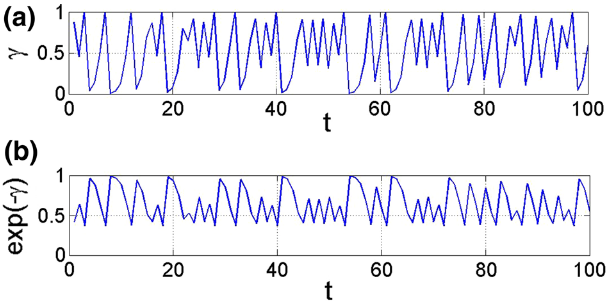


Fig. 8 Chaotic (Logistic) map for γ and it effect on β with $r = 1$ and $\beta_0 = 1$

using $\beta = \beta_{min} + (\beta_{max} - \beta_{min})e^{-\gamma r^2}$. In the standard firefly algorithm if $\gamma = 0$ then the attraction step length will be β_0 resulting any firefly can be attracted equally to any other brighter firefly irrespective of the distance between them and when r is very large with $\gamma \neq 0$ the step length vanishes. Hence the updating is equivalent with the standard firefly updating mechanism if $\beta_{min} = 0$, with $\beta_{max} = \beta_0$. However, this updating mechanism makes sure that the attraction step length is in the range $[\beta_{min}, \beta_{max}]$. Similar modification is presented in Meena and Chitra (2015), with β_{min} computed by the intensity difference of the two fireflies. The same updating equation for β is also used in Selvarasu and Kalavathi (2015), with β_{max} being limited between 0 and 1. Furthermore, β_{min} along with α and γ is encoded in the solution x_i, hence the dimension of x_i increases by three.

Other modification on the attraction is done by using different chaotic mappings as presented in Gandomi et al. (2013), where twelve different chaotic maps are presented to make γ and β adaptive. Similar work is done in Jansi and Subashini (2015) using chebyshev mapping and in AL-Wagih (2015), Abdel-Raouf et al. (2014) using sinusoidal map. The attraction term is supposed to be influenced by the light intensity and also the distance between the fireflies along with the light absorbtion coefficient. Furthermore, using a chaotic map for γ results a chaotic updating behaviour in β. For instance Fig. 8 a logistic map with a starting value of $\gamma = 0.87$ and its corresponding value of β with $\beta_0 = 1$ and $r = 1$. As can be seen from the figure when γ behaves in a chaotic way so does β. Hence, there is no need to update both γ and β at the same time. Rather making β_0 influenced by the light intensity at $r = 0$ and changing γ seems a better idea rather than updating β so that the influence of the light intensity will not be lost. In Coelho et al. (2011) a logistic map is used to update γ and in Long et al. (2015) β_0 is updated using gaussian map.

In Amaya et al. (2014), γ is modified to decrease with iteration using $\gamma = \gamma(1+K(\varepsilon-1))$, for a random number ε from a uniform distribution between 0 and 1 and K is a new parameter. It should be highlighted that K should be between zero and one, otherwise a negative value for γ may result. γ is modified in Coelho and Mariani (2013) using $\gamma = 0.03|G|$ where G is generated from a normal distribution given by $N(0, 1)$ with $\beta_0 = 1$, which limits the value of γ in a certain region. Theoretically, γ can have any positive value. A similar modification with additional iteration term added is given in Coelho and Mariani (2012) using $\gamma = |G|x'\frac{t}{t_{max}}$ where G is generated from a normal distribution given by $N(0, 0.3)$ and x' is the scaled value of x using linear scaling in such a way that it will put x' in between 0 and 1. The updating gives a direct relation between γ and number of iteration, t, that results a decrease in step length of attraction. There is also additional modification of γ as a function of iteration number is given in Liu et al. (2015), Fu et al. (2015). The update formula for γ given in Liu et al. (2015) is given by $\gamma = \gamma_{min} + (\gamma_{max} - \gamma_{min})\frac{t}{t_{max}}$. It is a linearly increasing function starting from

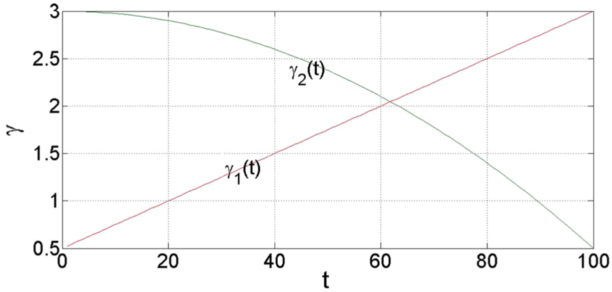


Fig. 9 $\gamma_1(t)$ is from Liu et al. (2015) and $\gamma_2(t)$ is from Fu et al. (2015)

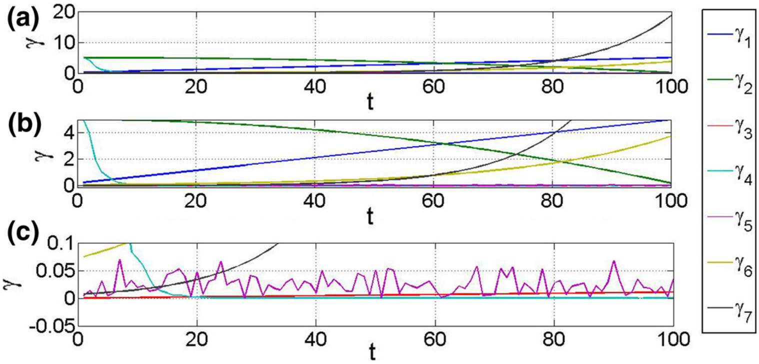


Fig. 10 Different scenarios of adaptive γ where γ_1 is from Liu et al. (2015); γ_2 is from Fu et al. (2015); γ_3 is from Coelho and Mariani (2012) with x generated randomly between zero and one; γ_4 is from Amaya et al. (2014) for $K = 0.5$; γ_5 is from Coelho and Mariani (2013); γ_6 is from Lukasik and Zak (2009) with $\omega = 1$, $\gamma_0 = 0.75$ and r_{max} made to decrease from $rmax$ starting from 10 decreases exponentially to 0.2 in the 100th iteration using $r_{max} = 10.4029e^{-0.0395t}$ and α_7 is the same with α_6 except $\omega = 2$. Furthermore, $\gamma_{max} = 5$ and $\gamma_{min} = 0.2$ is set to be used whenever needed

$\gamma_{min} + \frac{\gamma_{max}-\gamma_{min}}{t_{max}}$ at the first iteration, $t = 1$, and γ_{max} in the final iteration. This shows that the attraction decreases with a reasonable change of r. However, in Fu et al. (2015), a decreasing updating formula is propose using $\gamma = \gamma_{max} - (\gamma_{max} - \gamma_{min})(\frac{t}{t_{max}})^2$, with a final value being α_{min}. Figure 9, shows the two approaches with $\gamma_{min} = 0.5$ and $\gamma_{max} = 3$. Another modification based on γ is given in Lukasik and Zak (2009). γ is set to decrease based on the maximum distance between fireflies using $\gamma = \frac{\gamma_0}{r^{\omega}_{max}}$ for $\omega = 1, 2$ for γ_0 being between 0 and 1. Hence, the square of the distance expression in β will be minimized (as the iteration increases both r_{max} and r decreases, resulting r' more or less equivalent to a neighborhood of a constant number smaller than 1), and β will be independent of the distance, resulting all fireflies are viable to any other firefly, for $\omega = 2$. If $\omega = 1$ the effect of the distance will not be in squares but will still be there. The adaptive scenarios for γ is summarized in Fig. 10. As can be seen from Fig. 10, γ_5 is neither decreasing nor increasing, there are two decreasing scenarios, which are proposed in γ_2 and γ_4 . The one from γ_2 decreases slower than a linear function and also from the one in γ_4. The other four approaches gives an increasing α, γ_3 produced a very low values with slow increase compared to the other increasing scenarios; γ_7 grows slower than γ_6 until around $t = 60$, and start to grow quickly and over take even γ_1 around $t = 80$. γ_1 grows quicker than γ_6.

In Amaya et al. (2014), Wang et al. (2014b), Tilahun and Ong (2012c), modifications are done based on the light intensity difference between the fireflies. Suppose firefly j is brighter than firefly i, then for firefly i to move towards firefly j, ξ needs to be calculated using $\xi = \frac{I_j - I_i}{I_{max} - I_{min}}$ which always put ξ in between zero and one. Then based on ξ, β_0 will be $\beta_0 = \{ \begin{matrix} \xi & \xi > \eta \\ \eta & \xi \leq \eta \end{matrix}$ where η is a new algorithm parameter, (Wang et al. 2014b). Bigger value of η results β_0 to be more random, rather than to be dependent on difference in light intensity. Hence, smaller value of η should be used, so that the effect of the brightness will be in the step length of the attractiveness move. In Maidl et al. (2013), a scaling parameter a is added. Hence, $\beta = a\beta_0 e^{-\gamma r^2}$, where $a = \frac{f(x_j) - f(x_b)}{max\{f(x)\} - min\{f(x)\}}$ for a current global best solution x_b, which depends solely on x_j, not x_i i.e. no effect of the performance of x_i is added. The scaling value is always in between 0 and 1. In Tilahun and Ong (2012c), β_0 is expressed based on light intensity of firefly i and firefly j, using $\beta_0 = e^{I_{0,j} - I_{0,i}}$ where $I_{0,j}$ and $I_{0,i}$ are the light intensity of firefly j and i at $r = 0$, respectively. Here, $\beta_0 > 1$, because firefly j is brighter than firefly i. Hence, for problems with big functional value compared to the size of the feasible region, the attraction step length will be too big, hence appropriate scaling parameter needs to be used.

Based on the problem properties γ was made adaptive with $\beta_0 = 1$, in Othman et al. (2015). Another modification is done in Cheung et al. (2014), which is given by $\beta_0 = \frac{d(x_i, x_b) - \min_j\{d(x_i, x_j)\}}{-\max_j\{d(x_i, x_j)\} - \min_j\{d(x_i, x_j)\}}$. Hence, β_0 is between 0 and 1. Furthermore, $\gamma = \frac{1}{1 + \sigma e^{-\rho\beta_0}}$, for new algorithm parameters σ and ρ. Parameter assignment is not an easy task for a user, hence expressing γ with two other parameters doesn't seem reasonable. Furthermore, β_0 is not expressed in terms of light intensity but based on its location compared with other fireflies. Hence, two fireflies with different intensity at $r = 0$ may have same β_0 value and that is against the original idea in the standard firefly algorithm.

In Lin et al. (2013), the attractive step length is modified and expressed using $\beta = \beta_0\gamma(1 - r')$, where $r' = \frac{r}{r_{max}}$, for $r_{max} = \max_{i,j}\{d(x_i, x_j)\}$. The effect of the distance decreases linearly.

In addition to γ and β_0, the way the distance is measured affects the attraction step length, β. Cartesian distance is used in most of the applications with a number of exceptions. In fact any function, satisfying the properties of a distance function, can be used to compute the distance between two fireflies. The distance representation has been normalized in Lin et al. (2013) using $\frac{r - r_{min}}{r_{max} - r_{min}}$ with $r_{min} = 0$ and $r_{max} = \sqrt{\sum_{d=1}^{n}(\max_i\{x_i(d)\} - \min_i\{x_i(d)\})^2}$. The new distance will be normalized and is between zero and one. In Yan et al. (2012), the parameter m given in the general form as in Eq. (5) is replaced by a function of the dimension of the problem and the the size of the feasible region by $r^m = r^{\eta\sqrt{nR}}$, where n is the dimension of the problem and R is the maximum range given by $R = \max_i\{x_{max}(i) - x_{min}(i)$ for all dimension $i\}$. A minimum variation is used in place of cartesian distance in Sulaiman et al. (2012). In Othman et al. (2015), the distance is defined based on cartesian distance formula and the property of the problem. Another modification in the distance computation is done in Subramanian and Thanushkodi (2013). In Subramanian and Thanushkodi (2013), instead of computing the distance on the feasible region it is computed in the outcome or functional space using $f(x_j) - f(x_i)$. It should be noted that the solutions are exploring the feasible region and hence, two solutions with big difference in functional value may in fact be close in the feasible region and viceversa. Hence, in general it can be misleading to use the distance on the solutions space.

3.2 Strategy level modifications

In this modification category four kinds of modification will be discussed.

3.2.1 Modifying the movement of best or worst solution

The performance of firefly algorithm highly depends on the updating strategy used for the brightest firefly, x_b. If it is allowed to move randomly with a big step length, it may move to a non-promising area and its performance decreases. Hence, the already achieved good solution, in a promising region, may get lost as memory is not utilizing in the standard firefly algorithm to track the previous position and performance. Hence, in Tilahun and Ong (2012c), the random movement of the brightest firefly is updated by choosing a best direction, a direction which improves the solution's performance if it goes in that direction, from randomly generated m_r directions, where m_r is a new algorithm parameter. If any of these directions are not improving then it will stay in its current position. In that way the best solution found will not get lost. Another similar modification is done in Verma et al. (2016), where the brightest solution check the component of each of the other solution for improvement. Hence, the update is done using

$y = x_b$
for i =1 : n(for all dimension)
  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

Both approaches make sure the brighter solution will be replaced by a better solution or will not change if better solution is not archived. In the first case a new solution in the neighborhood is searched whereas in the second approach the brighter solution is copy some components from the existing value and the result may not necessarily be in the neighborhood. In Fu et al. (2015), Gauss distribution based move is applied for the brighter firefly x_b, which is given by $x_b := x_b + x_b N(\mu, \sigma)$. This will be applied if the variance of the solutions before a predetermined M iterations is less than a given precision parameter η. Perhaps modifying the movement based on the performance of the algorithm is a good idea to escape local solutions. However still like in the standard firefly algorithm the best solution may get lost in the process, unless methods discussed on Tilahun and Ong (2012c), Verma et al. (2016) on the movement of the brightest firefly is used or a memory is utilized to keep the global best solution.

The updating of the brightest firefly along with other top fireflies are modified in Kazemzadeh-Parsi (2014, 2015), Kazemzadeh-Parsi et al. (2015) . Two ways are proposed. The first one is the top m_{top} solutions in an iteration will pass to be kept in memory and will replace weak solutions in the next iteration. This implies that the selected top fireflies will repeat the same search paradigm as in the previous iteration which possibly gives another m_{top} solutions resembling solutions in the current solution. The other approach is no update will be done on the top solutions. In both cases the ratio of the top solution selected matters on the performance of the solution and in general if the number of solution is larger than the number top fireflies, $N >> m_{top}$, then it can be considered as an improved version based on elitism concept. However if $N >> m_{top}$ is not true then rather than using solutions to explore and exploit we are just keeping them and wasting the computational power of the algorithm.

In the standard firefly algorithm the worst solution will be attracted to all other fireflies as it is the dimmer and has a least brightness. However, in Yu et al. (2015b) a new updating mechanism based on opposite numbers is introduced. The new updating formula for the worst solution, x_{worst}, is given by $x_{worst} := \{ \begin{matrix} x_b & \varepsilon < p \\ x_{max} + x_{min} - x_{worst} & otherwise \end{matrix}$, for a new parameter p. It should be noted that, larger value for p means the worst solution is likely to be put in the best solutions place and it decreases the search capacity of the algorithm from N fireflies to $N - 1$ as that location is already being found and being exploited by x_b. However, assigning smaller value for p results a change of x_{worst} by its opposite number. Hence, using a smaller value for p gives better exploration.

3.2.2 Adding mutation operators

In order to escape from local solutions as well as improve the search mechanism beyond the vicinity of the existing solutions different modifications are proposed. These modifications are done by incorporating a mutation operator, crossover operator, by simple relocate a solution or by generating a random and feasible new solution.

In Kazemzadeh-Parsi (2014, 2015), Kazemzadeh-Parsi et al. (2015) , a random generation of new solutions is used to replace weak solutions. In the proposed modification ν random feasible solutions are generated and added to the solutions and the weak ν solutions will be deleted. In Shakarami and Sedaghati (2014), Olamaei et al. (2013), for each firefly x_i, three fireflies, x_1, x_2 and x_3, different from x_i will be selected randomly from the existing fireflies to produce a new test firefly, x_T, given by $x_T = x_1 + \varepsilon(x_2 - x_3)$. Then based on x_T, two new solution will be generated by $x_{m_1}(k) := \{ \begin{matrix} x_T(k) & \varepsilon_1 < \varepsilon_2 \\ x_b(k) & otherwise \end{matrix}$ and $x_{m_2} = \varepsilon x_b + \varepsilon(x_b - x_i)$, where ε, ε_1 and ε_1 are random numbers between 0 and 1. The better performing solution from x_i, x_{m_1} and x_{m_2} will be selected to be included in the solution set. Similar modification is proposed in Mohammadi et al. (2013), Kavousi-Fard et al. (2014) where for each x_i seven new solutions are generated based on the randomly selected three solutions, the best solution and the worst solution. Another similar mutation operator for using firefly algorithm for multiobjective optimization problem is proposed in Amiri et al. (2013). In all of the mentioned cases, for each solution x_i, new solutions will be generated using different but more or less similar operators and the better solution found will replace x_i. It should be noted that the main motivation of metaheuristic algorithm is that exhaustive search is not always possible rather a systematic exploration along with exploitation techniques are used. A mutation operator can enhance the standard firefly algorithm by giving a diversify solution and increasing its exploration property. However, generating many solutions in using a mutation operator resembles as if exhaustive search is being conducted and it may increase the complexity of the algorithm significantly for specially high dimensional problems. Hence, it needs to be wisely implemented.

In Wang et al. (2012), a portion of fireflies with better brightness are selected to be in top firefly list. For each firefly in the top group another firefly from the top fireflies list will be chosen at random and a new solution on the line joining them and near to the brighter one will be computed to replace one of these solutions, if its performance is better. This makes a focus on exploitation, and if the two solutions are in the neighborhood of the same local solution, both will not scape the local solution using this updating operator.

Another modification which can be mentioned in this category is a modification proposed in Bidar and Kanan (2013). The performance of each of the solutions will be recorded and if a hazardous condition, a condition which shows the performance of a solution in previous iterations, is under a given threshold, which means if that solution is not performing well in previous iterations, then it will be relocated to a new solution. Saving the performance of each of the solutions may significantly affect the memory and algorithm complexity, especially for high dimensional problems with wide feasible area where the number of random solutions is expected to increase highly.

3.2.3 New updating formula

In some of the researches done to improve the performance of the standard firefly algorithm modification are done in the updating formula given in Eq. (6).

In Lin et al. (2013), a new updating formula $x_i := x_i + \beta(x_j - x_i)(\alpha\varepsilon)$ for a modified $\beta = \beta_0\gamma(\frac{r_{ij}}{r_{max}})$ for r_{max} being the maximum distance between fireflies, is given. Here, the random movement expression is multiplied with the attractiveness term with an objective of increase the intensification. However, it simply omits the random movement with a new $\beta = \beta\alpha\varepsilon$. This means the random term simply affected the attractiveness step length, hence a firefly follows brighter firefly and will not move in a random movement. That is a huge drawback from the exploration point of view. A similar modification is proposed in Hassanzadeh and Kanan (2014) using $x_i := x_i + [\beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \sum_{h=1}^{k} A(h)\beta_0 e^{-\gamma r_{ih}^2}(x_h - x_i)]\alpha(\epsilon() - \frac{1}{2})$, where $A(h) = \frac{f(x_b)}{l(f(x_h) - f(x_b))}$ for l being a new algorithm parameter. The authors started with a wrong claim saying "*in the standard firefly algorithm, only one firefly in each iteration can affect others and attract its neighbors*", and hence try to make a firefly to follow better k fireflies. In addition multiplying the attraction term with the random movement will only influence the step length of the attraction and omits the random movement, as discussed above.

Another modification which is given in Amaya et al. (2014), Azad (2011), modifies the update of a firefly x_i which is attracted to x_j by updating it based on the vicinity of the brighter firefly. For example in Azad (2011), the update formula used is $x_i := x_j + \beta(x_j - x_i) + \alpha(\varepsilon() - 0.5)$. This updating formula is equivalent with $x_i := \beta x_i + (\beta + 1)(x_j - x_i) + \alpha(\varepsilon() - 0.5)$ and the second term moves x_i towards x_j in a step length larger than the distance between them. That simply is a neighborhood search for x_j in a specific direction and solutions in between x_i and x_j will not have a chance to be explored. However in Amaya et al. (2014), the second term is omitted and the updating formula is given by $x_i := x_j + \alpha(\varepsilon() - 0.5)$ which is equivalent with $x_i := x_i + (x_j - x_i) + \alpha(\varepsilon() - 0.5)$. This means the attractiveness step length , β, is one. That means it will move directly to x_j and updated based on the random term. Like the first case this one also performs a neighborhood search around x_j. However, when comparing the two modifications, the first case is better as the step length for the neighborhood around x_j is larger because there is additional movement than the random movement.

The updating formula is modified by multiplying the first term by a decreasing weight term given by $x_i := wx_i + \beta(x_j - x_i) + \alpha(\varepsilon - 0.5)$ where $w = w_{max} - (w_{max} - w_{min})\frac{t}{t_{max}}$. β_0 is fixed to be 1, (Tian et al. 2012).

In Kazemzadeh-Parsi (2014, 2015), Kazemzadeh-Parsi et al. (2015), the authors claim that instead of a firefly to move step by step towards better or brighter fireflies, it can move towards a representative point, P, which shows the over all distribution of the brighter firefly. P is is the average location of all brighter fireflies compared to x_i. Based on this the updating formula will be $x_i := x_i + \beta(P - x_i) + \alpha(\varepsilon() - 0.5)$. In the standard firefly algorithm, if x_j attracts x_i then the brightness of x_j is used to determine the attraction step length. However, in this modification, the authors didn't mention if the brightness is also averaged or the brightest firefly's brightness is used or what other approach they used. In addition taking the average direction, results that the attraction will be the same irrespective of how strong the attraction is, hence, if many weak but better than x_i solutions exit in a ceratin neighborhood far from global solution while there is a far better solution relatively near x_i, x_i will be mislead to follow the other solution equally. Furthermore, the computational complexity still will be the same, if we move the solution step by step or computing a representative direction as computing a representative direction itself will be done step by step.

In Shafaati and Mojallali (2012), Hongwei et al. (2015) an updating of solutions with additional attracting towards the global best is given, i.e. in addition to attracting to brighter solutions and move randomly, another term represent an attraction to the global solution is given. Hence, a memory needs to be used to save and update the global solution. In similar manner, in Goel and Panchal (2014), the updating is made with additional two terms. The first is towards the global best, and the second is towards the best solution in that iteration. However, we would like to highlight that moving towards the brightest solution is already included in the second term of the updating equation in the standard firefly algorithm, and adding another attraction means increasing the step length and that may not always give us a result desired.

A modification on the random movement term is given in Yu et al. (2015) based on and to control the diversity of the solutions. The update of the solution will continue using the usual way if the diversity of the solution is acceptable, where the diversity given by $\frac{1}{NL}\sum_{i=1}^{N}||x_i - \bar{x}||$ for $\bar{x}$ is the average location of all fireflies, N is total number of solutions and L is the largest diagonal of the feasible region. If the diversity is below a given threshold, then a new updating formula based on $\bar{x}$ will be used. The new updating formula is given by $x_i := x_i + \beta(x_j - x_i) + \alpha\varepsilon(x_i - \bar{x})$. The intention of this updating mechanism is to diversify the solution set. The random movement term indeed takes the firefly away from the average location of all fireflies and hence the diversity may be achieved. However, it would be better if this term is implemented to some of the solutions with a certain criteria based on their performance, otherwise in some cases it may hinder the search behavior of the algorithm.

$x_i = \{ \begin{matrix} x_i + \beta_0 e^{-\gamma r^2}(x_j - x_i) + \alpha(\varepsilon() - \frac{1}{2})(x_{max} - x_{min}), & \varepsilon > \frac{1}{2} \\ \frac{t_{max}-t}{t_{max}}(1-\eta)x_i + \eta x_b, & otherwise \end{matrix}$ is another updating forumla where η is calculated based on grey relational analysis, is is proposed in Cheung et al. (2014). This means that in half of the time the usual updating formula will be used and in the rest using the new formula which is equivalent with $x_i := x_i + \eta(x_j - x_i) + \frac{t}{t_{max}}(\eta - 1)x_i$. Hence the attraction will be there with different step length and the random movement is modified in a certain direction. One of the weakness of this modification is the introduction of a number of new parameters.

After the update is done using the updating formula used in the standard firefly algorithm, additional updating equation called social behaviour, as the authors called, is introduced in Farahani et al. (2011a, b), Kanimozhi and Latha (2013). For each solution at the end of the iteration, it will be updated using $x_i := x_i + \alpha(\varepsilon() - 0.5)(1 - P)$ where P is a random number from gaussian distribution. This means x_i is updated using $x_i := x_i + \beta(x_j - x_i) + \alpha(\varepsilon() - 0.5) + \alpha(\varepsilon() - 0.5)(1 - P)$ which is equivalent with $x_i := x_i + \beta(x_j - x_i) + \alpha(\varepsilon() - 0.5)(2 - P)$. Hence, the new added mechanism is equivalent with updating the random movement using gaussian distribution.

In Arora and Singh (2014a, b), another modification is proposed. In the modification, the brighter firefly donates some of its features based on a new algorithm parameter called probability of mutation, p_m. It is not mentioned what features and the amount of the features copied from the brighter firefly. 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 Arora and Singh (2014b), this operator of coping features from the brighter firefly will replace the updating formula of the standard firefly algorithm whereas in Arora and Singh (2014a), it will be done after the update is taken place using the usual updating scheme. If the usual updating scheme is replaced, the random movement of the standard firefly algorithm will be omitted and that significantly can affect the exploration behavior of the algorithm. Perhaps it is better to add some randomized parameter with the new updating approach. The new approach works similar with the attraction term in the update formula of the standard firefly algorithm, as the distance between the two fireflies decrease resulting x_i approaches or moved to x_j, by copying some of its components from x_j.

3.2.4 Modification on the structure

In a modification presented in Fister et al. (2013a), each component of a solution, $x_i(k)$, will be represented by quaternion, $x_i(k) = q_i^{(k)} = (q_i^{(k)}(1), q_i^{(k)}(2), q_i^{(k)}(3), q_i^{(k)}(4))$, and the updating will be done over the quaternion space. In order to compute the brightness and measure the fitness of the solutions, the euclidian space is used by changing the quaternion space using a norm function, $x_i(k) = ||q_i^{(k)}||$. It is clear to see that the dimension of the problem will become four fold. However, since it zooms to each component and try to optimize, a better result can be achieved. It would be interesting to explore the search behavior mathematically.

An opposite based learning is used in Verma et al. (2016) to generating the initial solution. Like the standard firefly algorithm N random feasible solutions will be generated and their opposite number is computed using $x_{min} + x_{max} - x_i$ for each firefly x_i. Hence, the total number of solution will be $2N$. The best N solutions will be taken as initial solutions for firefly algorithm.

In Dhal et al. (2015b), starting from the first iteration a parent of N solution and the new updated N children will be merged to form a total of $2N$ solutions. Then if $\varepsilon < (0.5 - \varphi)$ a solution will be chosen from low performing solutions otherwise from top performer solutions, for φ measuring the diversity based on average, minimum and maximum fitness of the fireflies with φ approaching one implying good diversity.

In the standard firefly algorithm a firefly will move towards all brighter fireflies. However, in modifications done in Yan et al. (2012), Fateen and Bonilla-Petriciolet (2014) a firefly will move not to all brighter fireflies but to a portion of brighter fireflies. For instance, in Yan et al. (2012) a winking parameter is added so that a firefly, x_i, will be attracted to another brighter firefly, x_j, if x_j is winking. However, even though a firefly is attracted not to all better solutions, no winking method is added in Fateen and Bonilla-Petriciolet (2014), rather a fraction of the total population of top fireflies will be chosen and a firefly will move towards the fireflies in the top fraction if it is brighter than itself. In the modification done in Banati and Bajaj (2011) also, a firefly does not move to all brighter fireflies but by identifying the one which will improve its performance. However, it should be noted that decreasing the performance of a solution is sometimes not bad in order to escape local solutions.

In addition to the modification mentioned, another approach using multi swarm approach in which the fireflies are divided into sub-populations and firefly algorithm is applied on each

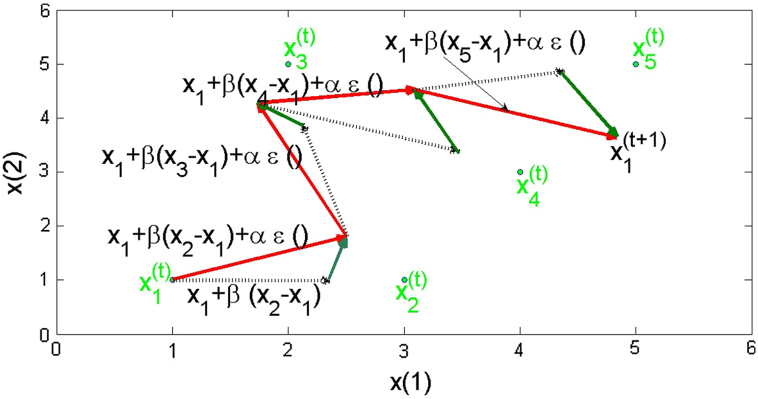


Fig. 11 The updating of x_1 towards four brighter fireflies in an iteration, the green arrow represent a random movement using $x_1 := x_1 + \alpha\epsilon$, the *broken arrow* represent an attraction movement and the *red arrow* represent the move by x_1 due to an attraction and a random movement, the resultant of the two. (Color figure online)

of these sub-populations for a fraction of the total iteration and with possible migration scheme between the sub-populations is proposed in Dugonik and Fister (2014). Furthermore, these multi swam approach is used for dynamic environment in Farahani et al. (2011c), Abshouri et al. (2011). Using different swarms as a sub-population has the advantage of exploring different promising region for multimodal problems. Furthermore, for high dimensional problems with wide feasible region or a complex problem with high computational complexity, a parallel implementation can easily be implemented using swarm sub-populations. Parallel implementation of firefly algorithm has already been done in some researches Wadhwa et al. (2014), Husselmann and Hawick (2011), Subotic et al. (2012), Paula et al. (2014).

4 Discussion

As any metaheuristic algorithms the performance of firefly algorithm depends on the appropriate tuning of the algorithm parameters. It basically has two parameters α and γ. α controls the step length of the random movement of a solution and γ controls the step length of the attraction term. Using the second term of equation (6) a firefly x_i moves towards other brighter firefly x_j with a step length $\beta = \beta_0 e^{\gamma r^2}$. Due to this attraction term x_i moves in the direction of x_j resulting an exploitation the area in between the two fireflies and the random movement plays as an exploration mechanism if the step length is sufficiently large. Figure 11 shows an iteration update of a solution x_1 towards other four brighter fireflies. However, keeping the step length large until the end of the solution procedure is not reasonable because the solution will jump around the approached solution rather than converging to the nearest optimal solution. So, most of the approaches proposed on α is making it adaptive to decrease through iterations (Subramanian and Thanushkodi 2013; Liu et al. 2015; Coelho and Mariani 2012; Yan et al. 2012; Goel and Panchal 2014; Shafaati and Mojallali 2012; Wang et al. 2012; Amaya et al. 2014; Shakarami and Sedaghati 2014; Olamaei et al. 2013; Kavousi-Fard et al. 2014; Brajevic and Ignjatovic 2015; Yang 2013; Manoharan and Shanmugalakshmi 2015; Baghlani et al. 2013; Fu et al. 2015; Yu et al. 2015a; Othman et al. 2015) As can be seen from Fig. 5, most of these algorithms decrease quicker than a linear function starting from the first iteration. However, starting to decrease the step length from the beginning will force the algorithm to converge to the nearest, possibly, local solution. Hence, in order to decrease the step length a certain criteria after a proper exploration of the solution needs to be set. It could be based on the performance of the solution. Some attempts are made in this regards (AL-Wagih 2015; Wang et al. 2014b; Yu et al. 2013), i.e. making the step length adaptive based on its performance in previous iterations. This means a possible modification, which will increase the step length and also decrease whenever necessary based on the solutions current situation and previous performance, is perhaps a good area to explore as a possible future work. A further research in this direction may possibly produce a promising result. In some researches, different probability distribution are also used (Farahani et al. 2011a, b; Liu et al. 2015; Brajevic and Ignjatovic 2015; Baghlani et al. 2013; Coelho and Mariani 2013; Amiri et al. 2013; Coelho et al. 2011; Fister et al. 2014; Dhal et al. 2015a; Yang 2010; Kanimozhi and Latha 2013). However, it would be interesting to study their strength and weakness based on a comparison using different class of problems.

In regard to the attraction movement, γ has been modified in a couple of papers. In the standard firefly algorithm β_0 is set to be one. Hence the step length depends the value assigned to γ. Small value for γ results larger step length of attraction. As can be seen from Fig. 10, different scenarios are presented in which γ increases, decreases or neither through iteration for a fixed β_0 (Liu et al. 2015; Coelho and Mariani 2012; Lukasik and Zak 2009; Amaya et al. 2014; Fu et al. 2015; AL-Wagih 2015; Coelho and Mariani 2013; Coelho et al. 2011; Selvarasu et al. 2013; Meena and Chitra 2015; Gandomi et al. 2013; Jansi and Subashini 2015; Abdel-Raouf et al. 2014; Long et al. 2015). Some researches proposed an adaptive attractiveness based on the light intensity of the solutions (Amaya et al. 2014; Wang et al. 2014b; Tilahun and Ong 2012c). Controlling the step length of the attraction based on the light intensity of the attractive firefly is the motivation given in the standard firefly algorithm and proposing an updating for β either β_0 or γ based on the current and possibly past performance of the solutions can be studied further.

The brightest solution needs to explore its neighborhood to improve its performance rather exploring other region otherwise it may lose its current value and deteriorate its performance. Hence, there are some researches proposed on how to update the brightest firefly (Tilahun and Ong 2012c; Verma et al. 2016). In addition, it is possible to keep the best solution in memory and go on with the usual updating strategy. Hence, some research utilize memory to save the performance of the solution to direct the updating strategy in future iterations. As far as the memory and time complexity is reasonable, it is a good idea to learn from experience. Promising results are proposed which keeps the best solution or update it in an 'accept only improving' way (Tilahun and Ong 2012c; Verma et al. 2016).

As can be seen from Fig. 11, the solutions tend to fly over the solution space together so the exploration property is smaller. In order to increase the exploration behavior by increasing α it will result the best solution to wonder around the solution space. Hence, perhaps different alpha can be used for the best solution and the others, as done in Tilahun and Ong (2014) or use a mutation operator to generate a solution possible away from the pack.

Proposing new modification formula which give a good exploration is an interesting area to explore and surely improve the premature convergence possibly towards local solutions. Some promising studies with mutation incorporation and proposed new formula to increase the exploration property of the algorithm are proposed (Shakarami and Sedaghati 2014; Olamaei et al. 2013; Kavousi-Fard et al. 2014; Amiri et al. 2013; Kazemzadeh-Parsi 2014, 2015; Kazemzadeh-Parsi et al. 2015; Mohammadi et al. 2013; Bidar and Kanan 2013; Yu et al. 2015). It should also be noted that for a solution if its performance decreases may not always a bad idea. For example for a misleading problem where the global solution is far from the packed local solutions, a solution needs to perform weak in order to reach to the global solution and move in a non promising direction to scape local solutions. Perhaps a modification on the updating strategy which incorporates this idea can be done in the future.

Another point that needs attention is the number of algorithm parameters. In order to assign a single parameter introducing more than one parameter may not always be reasonable. Some researches proposed a modification of the standard firefly algorithm, however a number of new parameters introduced and that by itself needs another study. Hence, the number of parameters should be put into consideration. Another issue in modifying firefly algorithm is to map the decision space or the feasible region to an 'easy to search' space. However, a single study, Fister et al. (2013a), is reported in this aspect and can be an interesting area for future work.

5 Simulation based comparison

A simulation based comparison of the modified versions along with the standard firefly algorithm is done. For the comparison purpose 14 modified versions from strategy level modification are selected along with the standard firefly algorithm. The criterion of selecting these algorithms are clear description which can be replicate for any problem and small number of additional parameters. These algorithms are; FFA1 from Tilahun and Ong (2012c), FFA2 from Verma et al. (2016), FFA3 from Yu et al. (2015b), FFA4 from Kavousi-Fard et al. (2014), Mohammadi et al. (2013), FFA5 from Shakarami and Sedaghati (2014), Amiri et al. (2013), FFA6 from Azad (2011), FFA7 from Amaya et al. (2014), FFA8 from Hongwei et al. (2015), FFA9 from Maidl et al. (2013), FFA10 from Goel and Panchal (2014), FFA11 from Yu et al. (2015), FFA12 from Farahani et al. (2011a, b), Kanimozhi and Latha (2013), FFA13 from Yang (2010) and FFA14 from Tian et al. (2012).

5.1 Benchmark problems

For the simulation purpose forty benchmark problems are used. The benchmark problems are constructed by varying the dimension of ten base problems.

1. First base problem: The first problem is Rastrigin function (Molga and Smutnicki 2016). It is a multimodal, continuous, differentiable and separable problem. It is given as in Eq. (8).

$$f_1(x) = 10D + \sum_{i=1}^{D}(x_i^2 - 10cos(2\pi x_i)) \tag{8}$$

   The feasible region is $-5.12 \leq x_i \leq 5.12, \forall i$. The optimum solution is found at $x_i^* = 0, \forall i$ with $f_1(x^*) = 0$.
2. Second base problem: The second problem is Alpine01 function (Gavana 2013). It is a multimodal, continuous, non-differentiable and separable problem, as given in Eq (9)

$$f_2(x) = \sum_{i=1}^{D} \mid x_i sin(x_i) + 0.1x_i \mid \tag{9}$$

   The feasible region is $-10 \leq x_i \leq 10, \forall i$. The optimum solution is found at $x_i^* = 0, \forall i$ with $f_2(x^*) = 0$.

3. Third base problem: The third problem is a multimodal, discontinuous, non-differentiable and separable problem (Tilahun et al. 2016; Tilahun 2017). It is given in Eq. (10).

$$f_3(x) = \sum_{j=1}^{D} \left[\sum_{i=1}^{5} p(i) \lfloor x_j \rfloor^{5-i} \right] \tag{10}$$

for $p = [0.03779 \quad -0.8405 \quad 6 \quad -14.42 \quad 7.134]$. The feasible region is $-1 \leq x_i \leq 12, \forall i$. The global optimum is found at $2 \leq x_i^* \leq 3, \forall i$ with $f_3(x^*) = -3.82536D$.
4. Fourth base problem: Ackley 2 Function is the fourth test problem (Jamil and Yang 2013). It is a unimodal, continuous, differentiable and non-separable problem given by Eq. (11)

$$f_4(x) = -200e^{0.02\sqrt{\sum_{i=1}^{D} x_i^2}} \tag{11}$$

The feasible region is $-32 \leq x_i \leq 32, \forall i$. The optimum solution is found at $x_i^* = 0, \forall i$ with $f_4(x^*) = -200$.
5. Fifth base problem: XinSheYang01 is the fifth problem selected (Gavana 2013). It is a multimodal, non-differentiable, separable and stochastic problem. It is given as in Eq. (12).

$$f_5(x) = \sum_{i=1}^{D} rand_i |x_i|^i \tag{12}$$

The feasible region is $-5 \leq x_i \leq 5, \forall i$. The optimum solution is found at $x_i^* = 0, \forall i$ with $f_5(x^*) = 0$.
6. Sixth base problem: Cosine Mixture Function is chosen to be the sixth problem (Jamil and Yang 2013). It is a multimodal, discontinuous, non-differentiable, separable problem given as in Eq. (13).

$$f_6(x) = -0.1 \sum_{i=1}^{D} cos(5\pi x_i) - \sum_{i=1}^{D} x_i^2 \tag{13}$$

The feasible region is $-1 \leq x_i \leq 1, \forall i$. The optimum solution is found at $x_i^* = 0, \forall i$ with $f_5(x^*) = -0.1D$.
7. Seventh base problem: The seventh problem is Schaffer's F6 Function (Dieterich and Hartke 2012). It is a multimodal, continuous, differentiable and non-separable problem. It is given in Eq. (14).

$$f_7(x) = 0.5 + \frac{sin^2\left(\sqrt{\sum_{i=1}^{D} x_i^2}\right) - 0.5}{(1 + 0.001 \sum_{i=1}^{D} x_i^2)^2} \tag{14}$$

The feasible region is $-100 \leq x_i \leq 100, \forall i$. The optimum solution is found at $x_i^* = 0, \forall i$ with $f_7(x^*) = 0$.
8. Eighth base problem: The eighth problem is a stochastic, multimodal, non-differentiable, separable and a stochastic problem as given in Eq. (15) (Gavana 2013).

$$f_8(x) = \sum_{i=1}^{D} rand_i |x_i - \frac{1}{i}| \tag{15}$$

where $rand_i$ is a random number between 0 and 1 from a uniform distribution. The feasible region is $-5 \leq x_i \leq 5, \forall i$. The optimal solution is found at $x^* = (1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{D})$ with $f_8(x^*) = 0$.

9. Ninth base problem: StyblinskiTang is selected to be the ninth problem (Gavana 2013). It is given in Eq. (16).

$$f_9(x) = \sum_{i=1}^{D}(x_i^4 - 16x_i^2 + 5x_i) \tag{16}$$

The problem is a multimodal, continuous, differentiable and separable problem. The feasible region is $-5 \leq x_i \leq 5, \forall i$. The optimal solution is found at $x_i^* = -2.903534018185960$ with $f_9(x^*) = -39.16616570377142D$.

10. Tenth base problem: Michalewicz is selected to be the tenth problem (Bingham 2016). It is multimoda, continuous, differentiable and separable problem given as in Eq. (17).

$$f_9(x) = -\sum_{i=1}^{D} sin(x_i) sin^{2m}\left(\frac{ix_i^2}{\pi}\right) \tag{17}$$

where $m = 10$ and $0 \leq x_i \leq \pi, \forall i$. The optimal solution depends on the dimension of the problem. For instnace if $D = 2, 5$ and 10, then $f(x^*) = -1.8013, -4.687658$ and -9.66015, respectively (Bingham 2016).

5.2 Simulation setup

The simulations are performed on $Intel\ Core^{TM}$ i3-3110M CPU @ 2.40 Ghz 64 bit operating system. MATLAB 7.10.0 (R2010a) is used for these simulations.

Maximum number of iteration is used as a termination criterion and it is set to be 100. In addition, 30 trials are conducted. The same initial solution are used for each versions of the algorithms in each simulation with the number of solutions being 100, 200 and 300 for the 3, 5, 8 and 12 dimensional problems. In addition, $\gamma = 1.2$ and α is set based on the size of the feasible region. It set to be 5, 2.5, 0.8, 8, 1.2, 0.25, 25, 0.25, 0.25, and 0.4 for problems from the first to the tenth, respectively. Furthermore, for FFA1 $m = 100$; for FFA3 $p = 0.5$; for FFA10 $\lambda = 1$ and for FFA11 the tolerance for diversification, tol, is set to be 0.01. To generate a Levy random direction the inverse erf function is used, i.e $\frac{1}{2(erfinv(rand(D,1)))^2}$.

5.3 Simulation results

Based on the simulation setup given in the previous section and running the algorithm 30 times the average and standard deviation of the best functional value and also the CPU time is recorded as given in Table 2.

To compare the results of the algorithms Friedman test is used (Villegas 2016). The rank of each of the algorithms in the 40 instances of problems is presented in Table 3.

The null hypothesis of the test is that there is no significant difference on the performance of the algorithms. The test statistics, T, can be calculated using Eq. (18) for P problems and K algorithms.

$$T = \frac{(P-1)[B - \frac{PK(K+1)^2}{4}}{A - B} \tag{18}$$

Table 2 Simulation results (T is CPU time, μ is mean value and SD is the standard deviation)

	D		FFA $f(x^*)$	T	FFA1 $f(x^*)$	T	FFA2 $f(x^*)$	T	FFA3 $f(x^*)$	T
f_1	3	μ	0.136291	4.634375	0.007429	4.890625	0.000002	5.907813	0.137170	4.584375
		SD	0.414339	0.480817	0.028689	0.403772	0.000020	0.447277	0.412641	0.305672
f_2	3	μ	0.784350	1.622410	0.008633	1.726931	0.000106	2.135654	1.092587	1.672331
		SD	0.454950	0.085445	0.005313	0.151436	0.000079	0.158315	0.583687	0.257473
f_3	3	μ	−10.434090	2.478856	−11.476080	2.489776	−11.476000	3.299421	−8.795860	2.374335
		SD	0.896802	0.184356	0.000000	0.148397	0.000000	0.162163	2.485837	0.127967
f_4	3	μ	−195.163361	3.096875	−199.536702	3.228125	−199.979279	3.940625	−192.705996	2.984375
		SD	1.617142	0.976081	0.227921	0.659433	0.010827	0.908994	3.603939	0.632679
f_5	3	μ	0.023602	2.051413	0.000263	2.257335	0.014634	3.102860	0.053689	2.226134
		SD	0.238055	0.297563	0.001263	0.173099	0.234365	0.307150	0.524482	0.303751
f_6	3	μ	−2.7	1.048438	−2.7	1.221875	−2.7	1.534375	−2.7	1.117188
		SD	0.000000	0.213154	0.000000	0.222171	0.000000	0.195406	0.000000	0.113214
f_7	3	μ	0.411565	3.6125	0.012699	3.692188	0.005629	4.175	0.352310	3.589063
		SD	0.068157	0.149289	0.006108	0.141883	0.001309	0.364494	0.074330	0.257594
f_8	3	μ	0.043827	1.192188	0.010453	1.298438	0.069543	1.575000	0.057509	1.098438
		SD	0.260853	0.173376	0.085003	0.102975	0.756436	0.198710	0.312159	0.166023
f_9	3	μ	−217.275099	0.973438	−234.811092	1.25625	−234.99658	1.517188	−205.225558	1.139063
		SD	10.214614	0.252918	0.155143	0.407663	0.000536	0.192842	17.349175	0.113501
f_10	3	μ	−2.696126	24.4402	−2.758071	24.7158	−2.760367	27.1962	−2.710571	23.7018
		SD	0.032794	0.1982	0.001776	0.1774	0.000046	0.0704	0.059102	0.2507
f_1	5	μ	0.344845	29.225000	0.061375	28.731250	0.000006	38.628125	0.421490	28.296875
		SD	0.416611	2.907229	0.1123886	4.803924	0.000001	6.477984	0.797152	3.527567
f_2	5	μ	0.050411	0.060178	0.010910	0.100121	0.000001	0.085431	0.039701	0.110302
		SD	0.013511	0.010051	0.005212	0.005246	0.000001	0.003354	0.015812	0.000271

Table 2 continued

	D		FFA $f(x^*)$	T	FFA1 $f(x^*)$	T	FFA2 $f(x^*)$	T	FFA3 $f(x^*)$	T
f_3	5	μ	−12.585401	2.70018	−17.135000	2.826891	−19.126001	4.542216	−13.675129	2.639150
		SD	3.65700	0.10011	1.854120	0.195020	0.000000	0.375921	2.924811	0.30010
f_4	5	μ	−166.457961	8.400120	−195.439765	8.789786	−199.978895	11.300010	−177.042241	8.100011
		SD	7.307201	2.468759	1.128600	2.345695	0.007600	2.900004	6.559094	2.146972
f_5	5	μ	0.047701	0.295223	0.000212	0.300101	0.004800	0.894450	0.013500	0.200010
		SD	0.037341	0.100213	0.000303	0.000322	0.010200	0.200123	0.012500	0.100222
f_6	5	μ	−4.5	35.603125	−4.5	36.665625	−4.5	50.715625	−4.5	34.325000
		SD	0	6.488699	0	7.287347	0	10.074081	0	7.946095
f_7	5	μ	0.491019	28.637500	0.032037	30.215625	0.006225	38.296875	0.468333	28.990625
		SD	0.004766	2.783606	0.005956	2.639085	0.000002	4.309909	0.027908	3.588168
f_8	5	μ	0.169621	37.553125	0.043171	37.681250	0.218472	50.943750	0.213515	37.300000
		SD	0.798181	4.795636	0.293512	3.788613	0.598989	8.327353	1.142126	5.686249
f_9	5	μ	−313.845959	27.759375	−389.488646	29.384375	−391.661647	37.737500	−308.984504	26.068750
		SD	38.217697	4.507150	0.664504	2.360942	0.000009	4.442468	32.631390	3.939710
f_10	5	μ	−2.621133	20.775000	−4.014620	21.203125	−4.350326	28.843750	−1.909591	20.612500
		SD	0.445160	8.691522	0.390609	8.417837	0.429383	10.501157	0.170497	7.827754

Table 2 continued

	D		FFA4 $f(x^*)$	T	FFA5 $f(x^*)$	T	FFA6 $f(x^*)$	T	FFA7 $f(x^*)$	T
f_1	3	μ	0	302.543750	0.044700	286.987500	0.068632	4.568750	0.012947	2.046875
		SD	0.000000	2.151714	0.099135	2.016100	0.266276	0.393303	0.067166	0.251191
f_2	3	μ	0.000000	42.805114	0.159987	34.000418	0.218933	1.556890	0.000667	0.728525
		SD	0.00001	3.584354	0.101445	2.005208	0.188110	0.093253	0.000245	0.048251
f_3	3	μ	−10.997070	111.518875	−11.476080	90.809742	−9.081030	2.394615	−9.081030	1.302608
		SD	1.009842	1.587198	0.000018	1.623334	0.523721	0.082630	0.100567	0.071772
f_4	3	μ	−200	183.640625	−198.366035	175.076563	−197.402667	2.420313	−199.975908	1.2484375
		SD	0.000000	1.647810	0.699129	3.967030	0.679007	0.417801	0.007244	0.249071
f_5	3	μ	0.000000	163.868130	0.002774	152.523738	0.003757	2.472616	0.000008	1.084207
		SD	0.000001	4.895246	0.017934	3.359346	0.038302	0.549545	0.000072	0.286262
f_6	3	μ	−2.7	47.959375	−2.7	42.829688	−2.7	0.629688	−2.7	0.298438
		SD	0.000000	0.750333	0.000000	0.223103	0.000000	0.106497	0.000000	0.055829
f_7	3	μ	0	182.117188	0.380975	172.854688	0.083471	2.0875	0.033080	1.04375
		SD	0.000000	2.540965	0.080583	2.220168	0.041712	0.207968	0.009747	0.360076
f_8	3	μ	0.004438	44.917188	0.016591	40.957813	0.024478	0.632813	0.001021	0.289063
		SD	0.031928	0.609622	0.106699	0.384151	0.120848	0.085977	0.016361	0.048440
f_9	3	μ	−234.90836	46.145313	−233.608536	41.609375	−231.477926	0.589063	−234.994889	0.267188
		SD	0.137453	0.409843	0.979220	0.426193	3.221314	0.094341	0.002435	0.027999
f_10	3	μ	−2.757243	2935.3200	−2.712017	2734.9107	−2.655310	26.9050	−2.755308	10.7901
		SD	0.004927	23.3191	0.052291	8.6150	0.074815	1.6897	0.000058	0.1171
f_1	5	μ	0.000000	2759.915625	0.183264	3098.221875	0.202197	40.865625	0.218901	16.315625
		SD	0.000000	515.597404	0.264871	875.429200	0.818026	10.154607	0.000326	4.122164
f_2	5	μ	0.000120	1.612034	0.011301	1.399985	0.027668	0.065755	0.000675	0.078856
		SD	0	0.185220	0.003803	0.095712	0.009100	0.042610	0.000226	0.030059

Table 2 continued

	D		FFA4 $f(x^*)$	T	FFA5 $f(x^*)$	T	FFA6 $f(x^*)$	T	FFA7 $f(x^*)$	T
f_3	5	μ	−17.977000	126.511562	−17.610001	102.53309	−15.323441	2.722270	−14.286099	1.722331
		SD	1.949890	47.043697	1.262891	43.100000	0.850859	0.300001	1.215091	0.211034
f_4	5	μ	−200.0	574.800112	−194.510111	493.556986	−193.950221	8.60000	−196.049573	3.890051
		SD	0.000000	96.467006	1.399303	80.099999	1.419300	2.503000	1.843800	1.266983
f_5	5	μ	0.000000	5.003211	0.000879	4.157992	0.001400	0.302201	0.000000	0.167795
		SD	0.000000	0.811453	0.000700	0.778969	0.000778	0.057761	0.000100	0.000012
f_6	5	μ	−4.5	3236.112500	−4.5	2906.481250	−4.5	36.318750	−4.374911	15.562500
		SD	0	1175.106415	0	1002.861653	0	5.144181	0.279708	1.617357
f_7	5	μ	0	2808.400000	0.376381	3013.650000	0.117057	35.7687500	0.116781	16.187500
		SD	0	507.891032	0.109158	819.991037	0.075022	6.687330	0.027428	3.529296
f_8	5	μ	0.007023	3043.225000	0.048694	2513.646875	0.059718	36.037500	0.007079	14.440625
		SD	0.042864	635.534290	0.260732	1026.022408	0.182331	3.921899	0.069105	1.964511
f_9	5	μ	−390.724837	1524.337500	−383.947551	1364.262500	−378.296284	24.128125	−391.661406	9.887500
		SD	0.585750	361.505658	3.511570	258.998828	4.873377	7.755664	0.000115	2.395889
f_10	5	μ	−3.827686	1651.087500	−3.273442	1363.371875	−2.924731	29.975000	−3.856894	12.575000
		SD	0.364889	338.159733	0.230110	351.991550	0.397767	3.161710	0.381085	2.364424

Table 2 continued

	D		FFA8 $f(x^*)$	T	FFA9 $f(x^*)$	T	FFA10 $f(x^*)$	T
f_1	3	μ	0.172268	23.201563	0.181008	0.548438	0.062878	19.726563
		SD	0.720280	1.570042	0.544757	0.031638	0.246919	1.060001
f_2	3	μ	1.347878	9.102658	1.533639	0.171601	0.264925	8.823417
		SD	0.803275	0.576348	0.527937	0.018013	0.118660	0.704032
f_3	3	μ	−8.210475	9.149459	−7.762634	0.352562	−10.78142	8.647135
		SD	1.485482	0.130779	2.150453	0.033860	0.896802	0.144304
f_4	3	μ	−186.708147	14.221875	−176.02110	0.43125	−180.775215	13.889063
		SD	4.828437	3.523243	8.133867	0.062152	9.873166	1.226857
f_5	3	μ	0.030449	11.899756	0.058611	0.336962	0.007604	15.016656
		SD	0.246042	1.885280	0.495669	0.150028	0.074614	3.205239
f_6	3	μ	−2.7	2.760938	−2.683554	0.248438	−2.696345	3.564063
		SD	0.000000	0.554635	0.035085	0.070822	0.011558	0.708580
f_7	3	μ	0.421343	13.63125	0.407897	0.392188	0.499769	14.259375
		SD	0.049586	3.333815	0.068567	0.099764	0.000353	0.779206
f_8	3	μ	0.043176	2.790625	0.067590	0.264063	0.052855	5.923438
		SD	0.231261	0.518128	0.462529	0.060043	0.339531	0.252209
f_9	3	μ	−178.222342	2.446875	−175.990294	0.220313	−190.288445	3.3875
		SD	18.679099	0.672912	26.903711	0.078454	19.599056	0.844097
f_10	3	μ	−2.716047	152.4494	−1.843541	0.6604	−2.573811	146.9165
		SD	0.014647	0.3362	0.11561	0.0238	0.269016	0.2565
f_1	5	μ	0.443117	156.425	0.437465	2.925000	0.134258	136.700000
		SD	1.161903	24.623053	1.295514	0.238096	0.348928	7.198560
f_2	5	μ	0.049887	0.087796	0.111499	0.000101	0.040300	0.076595
		SD	0.014932	0.042697	0.025111	0.008792	0.020311	0.022651

Table 2 continued

	D		FFA8 $f(x^*)$	*T*	FFA9 $f(x^*)$	*T*	FFA10 $f(x^*)$	*T*
f_3	5	μ	−12.232345	7.6654800	−10.004899	0.53799	−15.245397	9.365597
		SD	3.267401	0.677549	13.038889	0.122053	1.668788	0.896010
f_4	5	μ	−167.577785	29.401121	−104.374432	0.698877	−172.682234	64.220691
		SD	6.811600	8.500000	12.817890	0.088459	10.459000	18.100001
f_5	5	μ	0.042500	1.000023	0.095100	0.100001	0.001400	1.399999
		SD	0.037211	0.100014	0.091475	0.000010	0.001200	0.166799
f_6	5	μ	−4.5	155.534375	−4.443306	3.009375	−4.426177	135.875000
		SD	28.589691	0.065507	0.936301	0.082194	19.798827	0
f_7	5	μ	0.492806	147.112500	0.464038	2.743750	0.122775	134.190625
		SD	0.004054	19.055267	0.017558	0.279640	0.022690	6.439931
f_8	5	μ	0.242848	142.778125	0.223240	2.621875	0.043775	233.415625
		SD	1.495247	5.734347	1.436137	0.183685	0.090170	27.167145
f_9	5	μ	−300.326377	131.581250	−292.005510	1.840625	−358.217793	117.090625
		SD	39.913719	42.732615	35.872305	0.550812	12.756494	37.085908
f_{10}	5	μ	−3.716531	136.218750	−2.105569	2.668750	−2.110975	127.953125
		SD	0.426621	13.663158	0.278457	0.889212	0.321853	17.539621

Table 2 continued

	D		FFA11 $f(x^*)$	T	FFA12 $f(x^*)$	T	FFA13 $f(x^*)$	T	FFA14 $f(x^*)$	T
f_1	3	μ	0.172127	4.440625	0.209794	4.778125	0.308408	29.557813	0.033100	4.631250
		SD	0.720763	0.084882	0.610145	0.190628	0.842796	0.879732	0.111907	0.366897
f_2	3	μ	1.513693	1.694171	1.786221	1.859532	1.738999	24.489037	0.279522	1.670771
		SD	0.527264	0.188669	0.861742	0.255152	0.905186	2.378955	0.155767	0.187266
f_3	3	μ	−7.795054	2.433619	−8.351835	2.566217	−6.349073	22.879107	−4.099752	2.478856
		SD	2.241287	0.121061	1.073362	0.091038	2.853380	0.692645	5.227376	0.186253
f_4	3	μ	−186.565807	3.239063	−189.255333	3.593750	−173.085512	21.859375	−197.796637	2.915625
		SD	5.827818	0.443660	5.223245	0.606631	6.459846	5.669376	0.851731	0.712960
f_5	3	μ	0.038545	2.123174	0.055333	2.243294	0.089759	26.088047	0.006028	1.990573
		SD	0.460485	0.258484	0.384056	0.293588	0.464557	2.292879	0.041021	0.334682
f_6	3	μ	−2.7	1.078125	−2.7	1.231250	−2.7	6.2	−1.024539	1.014063
		SD	0.000000	0.066699	0.000000	0.040209	0.000000	0.256195	0.557321	0.164776
f_7	3	μ	0.399299	3.257813	0.447474	3.410938	0.497893	24.075	0.068031	3.479688
		SD	0.073782	0.344006	0.049860	0.618063	0.001433	0.601863	0.027053	0.427404
f_8	3	μ	0.097758	1.034375	0.065487	1.207813	0.241773	6.125000	0.023774	1.057813
		SD	0.660198	0.049301	0.357302	0.052109	1.520776	0.138389	0.156101	0.107763
f_9	3	μ	−178.101746	0.884375	−175.940634	1.045313	−104.288048	5.503125	−116.880255	0.95
		SD	18.985534	0.240623	14.0774000	0.325131	80.769784	1.432657	34.115551	0.249435
f_10	3	μ	−2.681226	31.5122	−2.649295	26.4422	−1.530193	420.6723	−1.559740	24.9810
		SD	0.029080	7.1744	0.065339	0.5440	0.456250	2.7302	0.859407	0.7857
f_1	5	μ	0.439414	33.575000	0.453604	33.609375	0.701889	217.746875	0.113372	30.653125
		SD	0.993352	1.664646	0.786943	1.249609	0.980294	22.634047	0.512501	2.501791
f_2	5	μ	0.051399	0.1000	0.049300	0.094677	0.097700	0.287595	0.010676	0.101510
		SD	0.012900	0.012001	0.014106	0.001253	0.031502	0.089127	0.003712	0.021331

Table 2 continued

	D		FFA11 $f(x^*)$	T	FFA12 $f(x^*)$	T	FFA13 $f(x^*)$	T	FFA14 $f(x^*)$	T
f_3	5	μ	−12.173741	2.73310	−12.865923	2.99021	−10.191789	28.724400	−9.488892	2.711000
		SD	3.391495	0.300301	4.070300	0.299935	12.914326	2.500000	11.436999	0.256977
f_4	5	μ	−165.940954	8.300124	−169.335994	8.409156	−148.589971	122.001287	−193.920651	8.511234
		SD	7.318300	2.300000	8.705100	2.355401	12.761567	30.199999	1.448500	2.456607
f_5	5	μ	0.047102	0.257787	0.059500	0.300000	1.472400	1.779855	0.001001	0.298879
		SD	0.032422	0.122001	0.045200	0.100000	2.461021	0.199999	0.000601	0.099879
f_6	5	μ	−4.5	33.081250	−4.5	35.184375	−4.5	260.378125	−3.849384	35.768750
		SD	5.269071	0	6.455011	0	61.833725	0.165243	6.816203	
f_7	5	μ	0.488345	32.871875	0.491618	32.653125	0.499119	210.218750	0.148216	31.112500
		SD	0.010094	1.367138	0.007660	1.475028	0.000214	18.939392	0.064647	2.755486
f_8	5	μ	0.186892	32.159375	0.193358	34.371875	0.304631	247.587500	0.056796	38.912500
		SD	0.661923	5.148745	1.030216	4.612584	1.095766	49.550998	0.198342	3.865264
f_9	5	μ	−295.156769	27.881250	−304.140889	30.306250	−282.694192	205.443750	−279.016100	31.000000
		SD	36.960362	5.693813	38.443241	4.578879	76.846959	22.692064	25.276563	2.031551
f_10	5	μ	−1.933519	30.031250	−2.418421	31.634375	−1.343842	163.231250	−0.242144	23.496875
		SD	0.414386	7.413746	0.391907	5.276678	0.437011	53.568453	0.270378	13.410772

Table 2 continued

	D		FFA $f(x^*)$	T	FFA1 $f(x^*)$	T	FFA2 $f(x^*)$	T	FFA3 $f(x^*)$	T
f_1	8	μ	0.819599	37.8125000	0.413651	37.325000	0.000208	44.725000	1.160345	37.140625
		SD	1.986987	10.154068	1.130387	8.814241	0.000077	9.942889	2.991857	8.198429
f_2	8	μ	0.875227	36.568750	0.259357	36.975000	0.000023	42.840625	0.871059	35.709375
		SD	2.517806	7.338459	0.830186	4.793013	0.000008	8.219161	2.642913	5.074032
f_3	8	μ	−19.293184	38.615625	−23.896740	39.965625	−30.602880	59.487500	−16.180394	37.046875
		SD	1.668484	9.352164	1.071099	9.128159	0.000000	14.426091	4.947121	6.042816
f_4	8	μ	−150.757509	35.434375	−194.742261	36.918750	−199.982839	50.643750	−164.197243	37.190625
		SD	5.512101	6.382812	0.442324	9.531662	0.005359	14.636053	8.077316	11.805649
f_5	8	μ	0.071575	36.246875	0.001185	36.568750	0.010531	55.084375	0.219532	35.50625
		SD	0.389947	3.589185	0.012506	4.280774	0.142259	6.647658	0.405478	4.768578
f_6	8	μ	−7.192534	37.609375	−7.200000	39.512500	−7.200000	58.268750	−7.200000	40.343750
		SD	0.016695	8.810193	0.000000	9.219845	0.000000	14.192564	0.000000	9.503591
f_7	8	μ	0.498751	35.543750	0.151690	37.603125	0.012333	52.096875	0.498813	36.925000
		SD	0.000748	8.042849	0.027686	7.942954	0.000001	9.952741	0.000605	5.663263
f_8	8	μ	0.391934	36.118750	0.094284	36.621875	0.386733	46.006250	0.434304	35.815625
		SD	1.702504	6.759403	0.330873	7.156782	1.122341	6.640787	0.953231	4.537272
f_9	8	μ	−542.339268	38.712500	−576.738447	38.437500	−626.658550	52.321875	−490.267076	37.656250
		SD	43.796375	8.430629	9.996308	7.797822	0.000070	13.050073	45.048997	5.818189
f_10	8	μ	−4.502683	34.078125	−5.071712	35.265625	−6.571427	48.106250	−4.048846	34.203125
		SD	0.741299	6.034315	0.403838	4.323412	0.822683	7.925952	0.408623	6.598237
f_1	12	μ	1.370882	88.743750	0.574514	88.306250	0.000003	97.265625	1.533285	87.665625
		SD	1.846789	3.778450	0.728746	4.758917	0.000009	12.152208	1.447350	7.803515
f_2	12	μ	1.415857	89.193750	0.614532	82.112500	0.000024	97.456250	1.870524	82.400000
		SD	1.536848	5.000178	2.502421	13.755115	0.000039	7.476366	4.033514	7.211285

Table 2 continued

	D		FFA $f(x^*)$	T	FFA1 $f(x^*)$	T	FFA2 $f(x^*)$	T	FFA3 $f(x^*)$	T
f_3	12	μ	−30.067128	80.900000	−35.813486	83.984375	−45.904320	108.940625	−29.263684	84.125000
		SD	3.866271	12.066783	1.060402	7.044042	0.000000	8.796621	13.083196	7.038130
f_4	12	μ	−117.791134	82.478125	−191.788203	81.559375	−199.987452	92.200000	−152.736505	82.971875
		SD	7.812020	6.374910	0.631772	6.160466	0.002868	7.595827	8.685918	8.044246
f_5	12	μ	0.221610	85.800000	0.000196	86.512500	0.000852	108.468750	0.736884	85.368750
		SD	1.375539	8.673968	0.001527	8.628335	0.017257	9.709262	3.755949	8.423017
f_6	12	μ	−10.784547	86.712500	−10.8	86.078125	−10.8	104.400000	−10.8	86.228125
		SD	0.034554	7.587612	0	7.639650	0	9.789581	0	8.802095
f_7	12	μ	0.499676	84.453125	0.245197	84.887500	0.012936	99.028125	0.499599	88.303125
		SD	0.004196	16.396139	0.035753	13.144560	0.001347	7.051909	0.000269	4.243973
f_8	12	μ	0.971347	89.003125	0.232469	93.562500	0.634547	103.909375	0.865316	90.956250
		SD	1.582827	13.734568	0.867651	6.466079	3.585646	7.789539	3.347291	5.035099
f_9	12	μ	−704.417005	88.918750	−761.352345	89.846875	−939.987903	106.531250	−−677.239865	89.328125
		SD	35.654581	7.650950	6.835436	5.308447	0.000037	6.508802	81.516082	5.056106
f_10	12	μ	−3.213469	88.484375	−4.829817	88.037500	−8.681864	106.118750	−2.879928	88.000000
		SD	0.522474	5.026844	0.054250	5.308852	1.329054	5.395996	0.409287	3.259227

Table 2 continued

	D		FFA4 $f(x^*)$	T	FFA5 $f(x^*)$	T	FFA6 $f(x^*)$	T	FFA7 $f(x^*)$	T
f_1	8	μ	0.000000	1186.350000	0.280702	985.200000	0.634623	39.140625	0.191634	17.050000
		SD	0.000000	261.773994	1.055163	198.159896	0.453352	2.863577	0.082210	0.931059
f_2	8	μ	0.000000	1132.628125	0.303206	956.643750	0.365299	43.687500	0.304180	18.631250
		SD	0.000000	285.537848	0.521185	196.241380	1.691406	16.599939	0.600704	7.924668
f_3	8	μ	−23.896740	3660.246875	−20.344910	3121.284375	−21.365950	40.078125	−19.620216	17.418750
		SD	1.071099	1483.824720	2.703349	1119.751177	0.795673	15.345651	1.680624	7.367593
f_4	8	μ	−200	2899.256250	−190.953819	2392.068750	−193.094185	38.628125	−190.541186	16.278125
		SD	0	1225.746851	1.922677	1068.529790	2.105806	8.083831	1.318158	2.921361
f_5	8	μ	0.000000	3414.86875	0.007060	2794.09375	0.001567	33.478125	0.006025	14.28125
		SD	0	1455.282645	0.060099	1183.867288	0.009154	12.265225	0.061056	5.661621
f_6	8	μ	−7.200000	4082.356250	−7.200000	2674.5968750	−7.176150	34.250000	−6.861123	14.618750
		SD	0.000000	1557.862865	0.000000	602.914581	0.038455	6.045795	0.518662	2.268280
f_7	8	μ	0.000000	3567.943750	0.409080	2549.562500	0.227770	38.415625	0.431359	16.325000
		SD	0.000000	832.030352	0.042682	588.859014	0.056727	9.577835	0.033058	3.731781
f_8	8	μ	0.036481	3293.771875	0.152866	2471.865625	0.117185	39.456250	0.208628	17.487500
		SD	0.038867	871.881069	0.595324	298.027730	0.240645	2.432045	0.584783	2.815394
f_9	8	μ	−603.449722	3669.609375	−594.737849	2631.131250	−571.048235	38.015625	−567.966281	16.284375
		SD	2.422269	715.686874	49.175236	634.690686	13.569465	7.033793	22.368511	2.773567
f_10	8	μ	−5.151769	3793.115625	−4.672797	2800.050000	−4.642948	38.203125	−4.839899	15.881250
		SD	0.300052	1290.246929	0.391538	965.601591	0.463311	13.099822	0.413707	5.937958
f_1	12	μ	0.000000	2505.196875	0.795759	2295.915625	0.736484	90.056250	0.658878	41.262500
		SD	0.000000	165.766383	0.66318569	94.641983	0.833247	3.312069	0.256351	1.766848
f_2	12	μ	0.000000	2847.675000	0.750337	2521.753125	0.785033	90.243750	0.726637	40.128125

Table 2 continued

	D		FFA4 $f(x^*)$	T	FFA5 $f(x^*)$	T	FFA6 $f(x^*)$	T	FFA7 $f(x^*)$	T
		SD	0.000000	108.624339	0.968269	146.908760	2.344514	3.919113	0.681632	2.052396
f_3	12	μ	−38.719170	3432.390625	−30.356400	2811.681250	−32.598700	88.431250	−31.717224	37.934375
		SD	0.000000	126.955888	3.748829	84.469090	1.407719	5.086035	2.634079	2.019298
f_4	12	μ	−200	2384.743750	−180.699734	2129.856250	−189.389872	87.987500	−179.159934	38.537500
		SD	0	108.258811	0.808267	81.543023	0.914689	4.131821	3.847367	1.601947
f_5	12	μ	0.000000	3129.015625	0.003514	2603.946875	0.000292	88.528125	0.010784	38.337500
		SD	0.000000	100.942183	0.020017	66.624805	0.000874	3.352271	0.057347	2.369407
f_6	12	μ	−10.8	3560.978125	−10.8	2867.453125	−10.713393	82.862500	−9.706163	38.946875
		SD	0	135.542945	0	117.321871	0.032850	13.462057	0.420299	2.196705
f_7	12	μ	0.000000	2513.490625	0.400311	2258.843750	0.345490	87.162500	0.485923	39.971875
		SD	0.000000	122.933793	0.090999	75.673824	0.051251	2.768881	0.009293	0.993349
f_8	12	μ	0.069161	2500.175000	0.406473	2241.487500	0.168012	89.843750	0.304663	39.309375
		SD	0.190834	112.373790	1.278062	82.162429	0.412238	3.398339	1.308512	1.118110
f_9	12	μ	−779.373778	2937.187500	−705.696559	2463.575000	−726.683213	92.606250	−720.236733	40.778125
		SD	6.907115	141.943171	20.233582	76.528122	12.560863	3.780837	14.095802	2.205877
f_10	12	μ	−5.135213	3077.453125	−4.341644	2581.015625	−4.589207	88.806250	−5.764130	39.721875
		SD	0.399515	136.991246	0.497906	87.082907	0.756594	11.644623	0.000329	5.452626

Table 2 continued

	D		FFA8 $f(x^*)$	T	FFA9 $f(x^*)$	T	FFA10 $f(x^*)$	T
f_1	8	μ	1.312569	202.24375	1.033004	1.359375	0.194355	387.109375
		SD	1.700534	34.601535	3.154164	0.426336	0.610224	86.189282
f_2	8	μ	1.371600	210.981250	1.193585	1.140625	0.184813	176.918750
		SD	2.090614	100.437562	1.808749	0.246311	0.326677	36.997153
f_3	8	μ	−16.958070	166.062500	−18.681652	2.090625	−21.051038	159.643750
		SD	4.919547	45.307449	6.323108	0.839718	5.774798	37.659071
f_4	8	μ	−136.659733	170.168750	−129.427922	1.571875	−191.646420	152.050000
		SD	6.417991	31.297617	7.790857	0.643913	0.927394	35.610166
f_5	8	μ	0.623584	161.781250	1.040704	1.75625	0.001415	153.040625
		SD	7.611981	64.592706	7.645346	0.925417	0.004821	65.293180
f_6	8	μ	−7.192736	156.668750	−6.571828	1.365625	−6.974842	140.703125
		SD	0.016243	24.433082	0.299983	0.172796	0.068497	29.859749
f_7	8	μ	0.498790	162.740625	0.497951	1.421875	0.231467	328.115625
		SD	0.000312	28.960214	0.000907	0.627631	0.030819	67.736123
f_8	8	μ	0.579614	202.193750	0.411856	1.259375	0.102570	178.309375
		SD	1.117555	37.325304	1.746519	0.289506	0.322837	42.414254
f_9	8	μ	−455.052634	178.528125	−483.282615	1.690625	−524.478230	155.137500
		SD	69.581195	36.702194	78.974261	0.718359	29.993751	25.576519
f_10	8	μ	−4.336500	182.328125	−4.474744	1.528125	−3.787257	349.556250
		SD	0.870594	43.827992	0.402707	0.355344	0.366459	84.080559
f_1	12	μ	1.629425	395.687500	1.434931	1.606250	0.541170	936.378125
		SD	1.007149	19.172477	0.703572	0.162905	0.764798	31.108936
f_2	12	μ	2.281231	396.265625	1.946232	1.568750	0.439465	954.534375

Table 2 continued

	D		FFA8 $f(x^*)$	T	FFA9 $f(x^*)$	T	FFA10 $f(x^*)$	T
		SD4.359453	18.186158	2.308709	0.042216	0.364416	45.290777	5.569224
f_3	12	μ	−27.032324	398.528125	−29.136568	2.203125	−30.929488	364.162500
		SD	8.307806	19.362132	13.970859	0.193901	9.228488	29.181009
f_4	12	μ	−106.152597	390.706250	−103.443981	1.434375	−188.678207	377.537500
		SD	7.214690	17.415636	9.163997	0.141663	1.452941	17.208487
f_5	12	μ	1.948956	395.437500	1.513560	1.853125	0.000221	375.290625
		SD	31.027994	17.132608	12.486134	0.091510	0.001331	15.866165
f_6	12	μ	−10.8	394.934375	−9.734409	1.537500	−9.282807	374.634375
		SD	0	25.051101	0.471740	0.128182	0.483635	33.537782
f_7	12	μ	0.499639	466.865625	0.499406	1.815625	0.328822	1046.343750
		SD	0.000121	30.436615	0.000184	0.168431	0.010834	50.793285
f_8	12	μ	0.887756	403.131250	0.917392	1.500000	0.183227	981.806250
		SD	1.704489	12.666068	1.678591	0.123031	0.177618	33.495067
f_9	12	μ	−604.777579	415.821875	−672.444051	2.068750	−691.606872	397.753125
		SD	78.406519	15.571113	110.734516	0.124805	57.571720	15.673423
f_10	12	μ	−2.974536	406.790625	−3.584349	2.143750	−2.825693	386.496875
		SD	0.538912	42.464419	0.362614	0.201132	0.390096	29.075899

Table 2 continued

	D		FFA11 $f(x^*)$	T	FFA12 $f(x^*)$	T	FFA13 $f(x^*)$	T	FFA14 $f(x^*)$	T
f_1	8	μ	1.126929	40.537500	1.065921	43.278125	1.184277	289.709375	0.208829	37.428125
		SD	4.572665	17.918280	2.968792	19.336073	2.742560	117.442262	0.876137	9.507576
f_2	8	μ	1.262516	40.959375	1.385928	41.450000	1.403281	295.906250	0.236157	38.328125
		SD	1.110698	9.759439	0.850682	9.637962	3.173793	75.821855	0.245204	5.381152
f_3	8	μ	−13.347444	38.265625	−11.059678	37.809375	−11.041358	263.100000	−10.124326	36.850000
		SD	4.818359	8.063583	4.968355	8.804777	8.551714	59.170493	8.142886	6.795902
f_4	8	μ	−143.213516	33.975000	−136.692637	34.625000	−129.752232	254.253125	−192.450834	34.140625
		SD	10.054519	10.412434	7.638194	10.949430	8.734697	65.331947	0.939054	10.404889
f_5	8	μ	0.314609	36.437500	0.665977	35.856250	1.340456	240.503125	0.001243	33.965625
		SD	0.763459	17.279512	5.028118	14.998325	13.207018	57.761702	0.005139	4.423799
f_6	8	μ	−7.200000	36.443750	−7.194166	36.259375	−7.200000	262.128125	−5.989793	36.787500
		SD	0.000000	4.937087	0.013046	6.105374	0.000000	52.904192	0.735112	7.846338
f_7	8	μ	0.499057	34.743750	0.498661	34.490625	0.499764	267.715625	0.230777	38.478125
		SD	0.000468	5.537388	0.000698	5.290321	0.000048	60.956966	0.060124	9.908526
f_8	8	μ	0.556441	36.440625	0.568344	38.240625	0.837491	261.781250	0.104124	36.100000
		SD	0.221411	6.330617	2.114961	6.792362	2.367356	59.220287	0.195002	5.466803
f_9	8	μ	−459.381425	38.115625	−447.258481	39.106250	−409.439887	314.853125	−423.193081	39.431250
		SD	31.795014	5.706491	71.383343	5.129516	118.569528	58.918495	44.921960	9.162052
f_10	8	μ	−3.961347	37.093750	−3.936979	35.971875	−3.857378	243.456250	−3.406608	37.215625
		SD	0.246344	8.182638	0.315016	9.369884	0.445320	52.012606	0.217051	4.881986
f_1	12	μ	1.633171	88.884375	1.575049	89.365625	2.122020	614.171875	0.590124	92.965625
		SD	1.530222	4.117930	1.327746	1.735838	2.380149	20.359723	0.609479	4.484694
f_2	12	μ	2.008212	87.650000	2.073712	87.690625	2.909466	628.543750	0.434489	93.300000
		SD4.359453	4.529695	4.437117	5.932383	5.356079	29.169156	0.620858	4.380624	

Table 2 continued

	D		FFA11 $f(x^*)$	T	FFA12 $f(x^*)$	T	FFA13 $f(x^*)$	T	FFA14 $f(x^*)$	T
f_3	12	μ	−25.444530	85.281250	−27.064540	86.062500	−25.382660	592.178125	−26.942622	85.800000
		SD	14.020343	9.782417	16.470469	10.988675	23.549252	59.898659	10.685807	8.537439
f_4	12	μ	−105.963288	87.781250	−104.590924	85.068750	−103.615931	606.696875	−189.109972	87.659375
		SD	5.861300	4.428041	5.813503	3.615400	4.894030	31.054118	1.335116	6.002262
f_5	12	μ	2.078874	91.078125	2.319036	89.371875	3.634825	622.868750	0.000130	90.962500
		SD	105.720932	4.808199	112.241752	3.889784	1499.384362	36.698350	0.000778	5.653537
f_6	12	μ	−10.780418	87.806250	−10.8	90.612500	−10.8	612.859375	−8.297116382	90.568750
		SD	0.043786	12.279369	0	3.821685	0	41.972009	1.650964	2.438041
f_7	12	μ	0.499449	93.403125	0.499614	90.659375	0.499898	695.640625	0.347750	94.200000
		SD	0.000306	3.022648	0.000181	4.578584	0.000028	91.914914	0.018064	5.047799
f_8	12	μ	1.156066	96.475000	0.831517	96.271875	1.412686	656.940625	0.153404	96.875000
		SD	2.949644	5.362683	1.969751	3.448112	3.133340	22.998963	0.186490	5.604923
f_9	12	μ	−640.193557	94.678125	−582.177497	92.596875	−549.935107	615.068750	−571.023598	90.331250
		SD	59.414219	3.010897	125.166214	2.934265	95.546848	71.975478	50.415886	11.328316
f_10	12	μ	−2.940717	92.162500	−3.162667	91.156250	−2.637322	629.981250	−2.498586	90.309375
		SD	0.580070	6.950444	0.593074	6.998805	0.267538	35.127574	0.141864	5.355791

Table 3 Ranking of the algorithms (D is the dimension, f is the problem and r is a rank of an algorithm in the ith problem instance)

D, f	FFA	FFA1	FFA2	FFA3	FFA4	FFA5	FFA6	FFA7	FFA8	FFA9	FFA10	FFA11	FFA12	FFA13	FFA14
$3, f1$	9	3	2	10	1	6	8	4	12	13	7	11	14	15	5
$3, f2$	9	4	2	10	1	5	6	3	11	13	7	12	15	14	8
$3, f3$	6	1.5	3	9	4	1.5	7.5	7.5	11	13	5	12	10	14	15
$3, f4$	8	4	2	9	1	5	7	3	11	14	13	12	10	15	6
$3, f5$	9	3	8	12	1	4	5	2	10	14	7	11	13	15	6
$3, f6$	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	14	13	6.5	6.5	6.5	15
$3, f7$	11	3	2	7	1	8	6	4	12	10	15	9	13	14	5
$3, f8$	8	3	13	10	2	4	6	1	7	12	9	14	11	15	5
$3, f9$	7	4	1	8	3	5	6	2	10	12	9	11	13	15	14
$3, f10$	8	2	1	7	3	6	10	4	5	13	12	9	11	15	14
$5, f1$	9	3	2	10	1	6	7	8	13	11	5	12	14	15	4
$5, f2$	12	5	1	8	2	6	7	3	11	15	9	13	10	14	4
$5, f3$	10	4	1	8	2	3	5	7	11	14	6	12	9	13	15
$5, f4$	12	4	2	8	1	5	6	3	11	15	9	13	10	14	7
$5, f5$	12	3	8	9	1.5	4	6.5	1.5	10	14	6.5	11	13	15	5
$5, f6$	6	6	6	6	6	6	6	14	6	12	13	6	6	6	15
$5, f7$	12	3	2	10	1	8	5	4	14	9	6	11	13	15	7
$5, f8$	8	3	12	11	1	5	7	2	14	13	4	9	10	15	6
$5, f9$	8	4	1	9	3	5	6	2	11	13	7	12	10	14	15
$5, f10$	8	2	1	13	4	6	7	3	5	11	10	12	9	14	15
$8, f1$	9	7	2	13	1	6	8	3	15	10	4	12	11	14	5

Table 3 continued

D, f	FFA	FFA1	FFA2	FFA3	FFA4	FFA5	FFA6	FFA7	FFA8	FFA9	FFA10	FFA11	FFA12	FFA13	FFA14
$8, f2$	10	5	2	9	1	6	8	7	13	11	3	12	14	15	4
$8, f3$	8	2.5	1	11	2.5	6	4	7	10	9	5	12	13	14	15
$8, f4$	10	3	2	9	1	7	4	8	13	15	6	11	12	14	5
$8, f5$	9	2	8	10	1	7	5	6	12	14	4	11	13	15	3
$8, f6$	10	4	4	4	4	4	11	13	9	14	12	4	8	4	15
$8, f7$	11	3	2	13	1	7	4	8	12	9	6	14	10	15	5
$8, f8$	9	2	8	11	1	6	5	7	14	10	3	12	13	15	4
$8, f9$	7	4	1	9	2	3	5	6	12	10	8	11	13	15	14
$8, f10$	7	3	1	10	2	5	6	4	9	8	12	11	13	14	15
$12, f1$	9	4	2	11	1	8	7	6	13	10	3	14	12	15	5
$12, f2$	9	5	2	10	1	7	8	6	14	11	4	12	13	15	3
$12, f3$	8	3	1	9	2	7	4	5	12	10	6	14	11	15	13
$12, f4$	10	3	2	9	1	7	4	8	11	15	6	12	13	14	5
$12, f5$	9	3	6	10	1	7	5	8	12	11	4	13	14	15	2
$12, f6$	9	4.5	4.5	4.5	4.5	4.5	11	13	4.5	12	14	10	4.5	4.5	15
$12, f7$	14	3	2	11	1	7	5	8	13	9	4	10	12	15	6
$12, f8$	13	5	8	10	1	7	3	6	11	12	4	14	9	15	2
$12, f9$	7	3	1	9	2	6	4	5	12	10	8	11	13	15	14
$12, f10$	8	4	1	12	3	6	5	2	10	7	13	11	9	14	15
Average rank	9.1125	3.6	3.425	9.375	2	5.7125	6.1625	5.5125	10.825	11.8	7.5375	11.2375	11.275	13.65	8.775
Sum of rank($\sum_{i=1}^{40} r_i$)	364.5	144	137	375	80	228.5	246.5	220.5	433	472	301.5	449.5	451	546	351
Sum of rank square ($\sum_{i=1}^{40} r_i^2$)	3460.25	576	864.5	3678.5	240	1386.75	1650.75	1599.75	4954.5	5748	2749.25	5227.25	5308.5	7781.5	4049

where A is the sum of the square of the ranks of all the algorithms and problems, i.e. $A = \sum_{i=1}^{P} \sum_{j=1}^{K} r_{ij}^2$ and B is the sum of values produced by squaring the sum of the ranks for each algorithm divided by the number of the problems, i.e. $B = \frac{1}{P} \sum_{i=1}^{K} \left[\sum_{j=1}^{P} r_{ij} \right]^2$.

For our problem $A = 49274.5$ and $B = 41901.70625$. Therefore, the test statistics, $T = 18.52$. Hence, the null hypothesis is rejected as the test statistics is greater than 0.99 quantile of the F distribution with 14 and 546 degree of freedom. This implies that there are at least two algorithms with significant performance difference. In order to compare the performance of each of the algorithm a pair wise comparison needs to be performed. Hence, for two algorithm i and j, if the difference in their rank fulfills the condition given in equation (19) then the two algorithms have different performance.

$$|r_i - r_j| > t_{1-\frac{\alpha}{2}} \sqrt{\frac{2P(A-B)}{(P-1)(K-1)}} \tag{19}$$

where $t_{1-\frac{\alpha}{2}}$ is the $1-\frac{\alpha}{2}$ quantile of the t-distribution with $(P-1)(K-1)$ degrees of Freedom. For our case with $\alpha = 0.01$, $t_{1-\frac{\alpha}{2}} = 2.585$. Hence, $t_{1-\frac{\alpha}{2}} \sqrt{\frac{2P(A-B)}{(P-1)(K-1)}} = 84.9621$. Based on this value and the difference in sum of ranks given in Table 4, it is possible to see which algorithms performs in similar way and which algorithm outperforms the other. If the entry is greater than 84.9621 then the corresponding algorithm in the column performs better than the corresponding algorithm in the row and if the entry is less than −84.9621 then the algorithm corresponding to the row performs better than the corresponding algorithm in the column. If the entry is in the interval [−84.9621, 84.9621] then the performance of the two algorithms is the same.

A similar analysis can be done for the CPU time. The results are demonstrated on the Fig. 12.

Hence, based on the simulation result FFA4, FFA2 and FFA1 are the best performing whereas FFA13 is the algorithm with least performance. However, if we look at the CPU time analysis, FFA9 is the algorithm with smaller CPU time whereas FFA5, FFA4, FFA13 are time expensive algorithms. Hence, based on the resource availability and the sensitivity of the problem, a user can decide which version of firefly algorithm to implement. For instance, if the problem is very sensitive and sufficiently enough time can be given FFA4 will be the best choice whereas if a quick solution is needed with a reasonable solution perhaps FFA1 is the best choice as it is among the top performers in terms of efficiency and also not among the worst in terms of CPU time. FFA2 will be the next best choice after FFA1 as it needs slightly more time than FFA1 to run but among the top efficient versions of firefly algorithm.

5.4 Discussion

Based on the simulation result on the forty problems, we can roughly categorize the algorithms into five categories. The first one contains algorithms which are computationally expensive but effective versions of the algorithm. It includes FFA4, FFA2 and FFA5. The second category include those algorithms which are computationally expensive and not so effective compared to the others and it includes FFA13 and FFA8. The third category is when the computational time is smaller and effective algorithms, which includes FFA1 and FFA7. The fourth category includes FFA9, FFA12 and FFA11, where the computational time smaller and their performance is not better when compared with the others. The last category is a category in the middle both in the performance as well as running time. This category includes FFA, FFA3, FFA6, FFA10 and FFA14. It is summarized in the Fig. 13.

Table 4 Rank difference (each entry is the subtraction of the rank of the algorithm in the column from the rank of the algorithm in the row)

	FFA1	FFA2	FFA3	FFA4	FFA5	FFA6	FFA7	FFA8	FFA9	FFA10	FFA11	FFA12	FFA13	FFA14
FFA	220.5	227.5	−10.5	284.5	136	118	144	−68.5	−107.5	63	−85	−86.5	−181.5	13.5
FFA1	–	7	−231	64	−84.5	−102.5	−76.5	−289	−328	−157.5	−305.5	−307	−402	−207
FFA2	–	–	−238	57	−91.5	−109.5	−83.5	−296	−335	−164.5	−312.5	−314	−409	−214
FFA3	–	–	–	295	146.5	128.5	154.5	−58	−97	73.5	−74.5	−76	−171	24
FFA4	–	–	–	–	−148.5	−166.5	−140.5	−353	−392	−221.5	−369.5	−371	−466	−271
FFA5	–	–	–	–	–	−18	8	−204.5	−243.5	−73	−221	−222.5	−317.5	−122.5
FFA6	–	–	–	–	–	–	26	−186.5	−225.5	−55	−203	−204.5	−299.5	−104.5
FFA7	–	–	–	–	–	–	–	−212.5	−251.5	−81	−229	−230.5	−325.5	−130.5
FFA8	–	–	–	–	–	–	–	–	−39	131.5	−16.5	−18	−113	82
FFA9	–	–	–	–	–	–	–	–	–	170.5	22.5	21	−74	121
FFA10	–	–	–	–	–	–	–	–	–	–	−148	−149.5	−244.5	−49.5
FFA11	–	–	–	–	–	–	–	–	–	–	–	−1.5	−96.5	98.5
FFA12	–	–	–	–	–	–	–	–	–	–	–	–	−95	100
FFA13	–	–	–	–	–	–	–	–	–	–	–	–	–	195

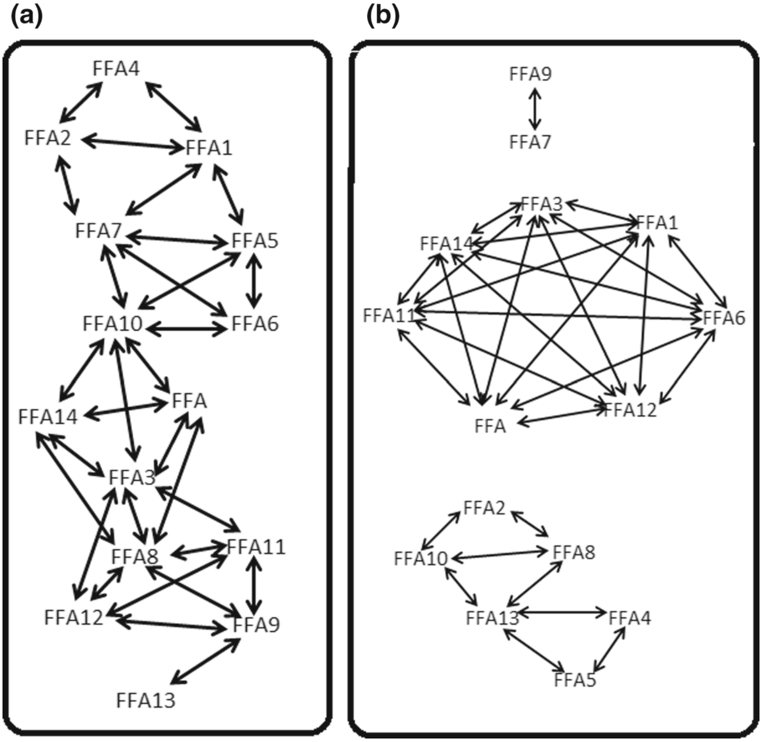


Fig. 12 The simulation result based on Friedman test. **a** is for the minimum functional value equivalence and **b** is CPU time equivalence, as going down the efficiency decrease and the *arrow* between the algorithm indicates that the two algorithms have similar performance

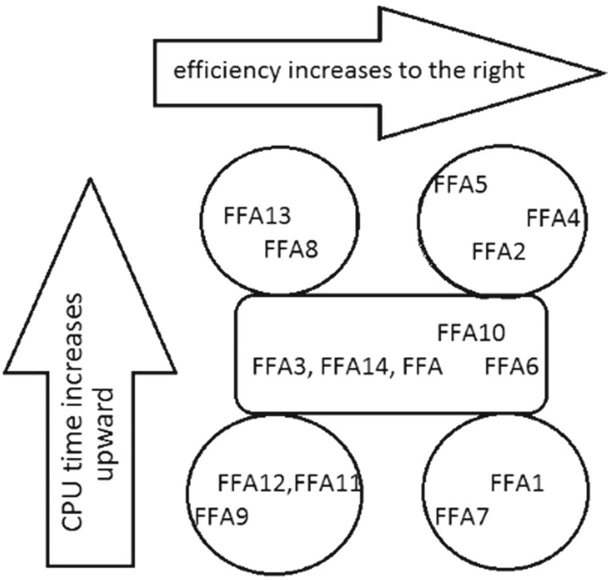


Fig. 13 Simulation result based sorting of the algorithm

If a sensitive problem is being solved with sufficient time then the algorithms in the first category are suitable whereas if a reasonable solution is needed within relatively short period of time the algorithms in the third category. The simulation is done based on a fixed parameter setting, which is based on recommendations from literature. It should be noted that by adjusting the parameters some of the algorithms can perform better. Hence, possible future works includes a detailed comparison with a wide number of problems and different parameter tuning. In addition the test problem used and their dimension is limited. The aim of this paper is to do a detailed theoretical analysis and have an initial simulation based comparison where advanced simulation based comparison is left for future works.

6 Conclusion

In this paper a detailed review of the standard firefly algorithm with its modified versions for continuous problems is discussed. Each of the modification is discussed and categorized based on two basic categories. The first category is parameter level modification where the algorithm parameters are modified to boost the performance of the algorithm. The step length for the random movement α has been made to decrease through iteration and different probability distribution is used. Decreasing of α is a good idea but the starting iteration in which α start decreasing or conditions needs to be studied. The same is done for γ with a decreasing, increasing and a neither increasing nor decreasing updating scheme, each with different strength, is proposed. In some research β_0 is made to be computed based on the light intensity of the fireflies and that is the original idea of the algorithm. The second category is strategy level category in which, modified movement for the best and worst solution, mutation incorporation, new updating formula and modifications on the structure is presented. In the movement modification of the brightest firefly, the best performance should not get lost. Hence, either a memory should be used to store the global best or only improving solutions should be accepted. Mutation operators are used to compute a new solution away from the swarm movement and that is one promising way of increasing the exploration property of the algorithm. Some research modified the updating formula in such away that a better exploration property is added to the algorithm. Change of search space to 'an easy to search' space and different probability distribution function to guide the random movement is also proposed. Each of the modified versions are presented and their weakness along with good aspects is reported. Possible future works are also outlined.

References

Abdelaziz AY, Mekhamer SF, Badr MAL, Algabalawy MA (2015) The firefly meta-heuristic algorithms: developments and applications. Int Electr Eng J (IEEJ) 6(7):1945–1952

Abdel-Raouf O, Abdel-Baset M, El-henawy I (2014) Chaotic firefly algorithm for solving definite integral, I.J. information technology and computer. Science 06:19–24

Abshouri AA, Meybodi MR, Bakhtiary A (2011) New firefly algorithm based on multi swarm & learning Automata in dynamic environments. In: Third international conference on signal processing systems (ICSPS2011), August 27Ű28, Yantai, China, 73–77, IEEE

Ali N, Othman MA, Husain MN, Misran MH (2014) A review of firefly algorithm. ARPN J Eng Appl Sci 9(10):1732–1736

Al-Wagih K (2015) Improved firefly algorithm for unconstrained optimization problems. Int J Comput Appl Technol Res 4(1):77–81

Alweshah M (2014) Firefly algorithm with artificial neural network for time series problems. Res J Appl Sci Eng Technol 7(19):3978–3982

Amaya I, Cruz J, Correa R (2014) A modified firefly-inspired algorithm for global computatiional optimization. DYNA 81(187):85–90

Amiri B, Hossain L, Crawford JW, Wigand RT (2013) Community detection in complex networks: multi-objective enhanced firefly algorithm. Knowl Based Syst 46:1–11

Ariyaratne MKA, Pemarathne WPJ (2015) A review of recent advancements of firefly algorithm: a modern nature inspired algorithm. In: Proceedings of the 8th international research conference, 61–66, KDU, Published November 2015

Arora S, Singh S (2014a) Performance research on firefly optimization algorithm with mutation. In: International conference on communication, computing & systems (ICCCS2014), 168–172

Arora S, Singh S, Singh S, Sharma B (2014b) Mutated fireïňĆy algorithm. In: International conference on parallel, distributed and grid computing, IEEE, 33–38

Azad SK (2011) Optimum design of structures using an improved firefly algorithm. Int J Opt Civil Eng 2:327–340

Baghlani A, Makiabadi MH, Rahnema H (2013) A new accelarated firefly algorithm for size optimization of truss structures. Scientia Iranica Trans A Civil Eng 20(6):1612–1625

Banati H, Bajaj M (2011) Fire fly based feature selection approach. IJCSI Int J Comput Sci Issues 8(4):473–480

Bidar M, Kanan HR (2013) Jumper firefly algorithm. In: Proceeding of international conference on computer and knowledge engineering (ICCKE 2013), Oct. 31–Nov. 01, 2013, Ferdowsi University of Mashhad, 278–282

Bingham D (2016) Virtual library of simulation experiments: test functions and datasets, 2015. http://www.sfu.ca/~ssurjano/michal.html. Accessed Feb 2016

Brajevic I, Ignjatovic J (2015) An enhanced firefly algorithm for mixed variable structural optimization problems. Ser Math Inf 30(4):401–417

Cheung NJ, Ding X-M, Shen H-B (2014) Adaptive firefly algorithm: parameter analysis and its application. PLoS ONE 9(11):1–12

Coelho LdS, Mariani VC (2012) Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. Comput Math Appl 64:2371–2382

Coelho LdS, Mariani VC (2013) Improved firefly algorithm approach applied to chiller loading for energy conservation. Energy Build 59:273–278

Coelho LdS, de A Bernert DL, Mariani VC (2011) A chaotic firefly algorithm applied to reliability-redundancy optimization. In: 2011 IEEE congress on evolutionary computation (CEC11), 517–521

de Paula LCM, Soares AS, Soares TWL, Delbem ACB, Coelho CJ, Filho ARG (2014) Parallelization of a modified firefly algorithm using GPU for variable selection in a multivariate calibration problem. Int J Nat Comput Res 4(1):31–42

Dhal KG, Quraishi MdI, Das S (2015a) A chaotic levy flight approach in bat and firefly algorithm for gray level image enhancement. I.J. Image Gr Signal Process 7:69–76

Dhal KG, Quraishi MdI, Das S (2015b) Development of firefly algorithm via chaotic sequence and population diversity to enhance the image contrast. Nat Comput. doi:10.1007/s11047-015-9496-3

Dieterich J, Hartke B (2012) Empirical review of standard benchmark functions using evolutionary global optimization. Appl Math 3:1552–1564

Dugonik J, Fister I (2014) Multi-population firefly algorithm. In: Proceedings of the 2014, 1st student computer science research conference, Ljubljana, Slovenia, 7 October 19–23

Farahani ShM, Abshouri AA, Nasiri B, Meybodi MR (2011a) An improved firefly algorithm with directed movement. In: Proceedings of 4th IEEE international conference on computer science and information technology, Chengdu, 248–251

Farahani ShM, Abshouri AA, Nasiri B, Meybodi MR (2011b) A Gaussian firefly algorithm. Int J Mach Learn Comput 1(5):448–453

Farahani SM, Nasiri B, Meybodi MR (2011c) A multiswarm basedfirefly algorithm in dynamic environments. In Third international conference on signal processing systems (ICSPS2011), August 27–28, Yantai, China, 68–72, IEEE

Fateen S-EK, Bonilla-Petriciolet A (2014) Intelligent firefly algorithm for global optimization. In: Yang X-S (ed) Cuckoo search and firefly algorithm, studies in computational intelligence 516, 315–330

Fister I, Yang X-S, Brest J, Fister I Jr (2013a) Modified firefly algorithm using quaternion representation. Expert Syst Appl 40:7220–7230

Fister I, Fister Jr I, Yang XS, Brest J (2013b) A comprehensive review of firefly algorithms, swarm and evolutionary computation. doi:10.1016/j.swevo.2013.06.001

Fister I, Yang X-S, Brest J, Fister Jr I (2014) On the randomized FireïňĆy Algorithm. In: Yang X-S (ed) Cuckoo search and FireïňĆy algorithm, studies in computational intelligence 516, 27–48

Fu Q, Liu Z, Tong N, Wang M, Zhao Y (2015) A novel firefly algorithm based on improved learning mechanism. In: International conference on logistics engineering, management and computer science (LEMCS 2015), 1343–1351
Gandomi AH, Yang X-S, Talatahari S, Alavi AH (2013) FireïňĆy algorithm with chaos. Commun Nonlinear Sci Numer Simulat 18:89–98
Gavana A (2013) Global optimization benchmarks and AMPGO. http://infinity77.net/global_optimization/test_functions_nd_X.html. Accessed Feb 2016
Goel S, Panchal VK (2014) Performance evaluation of a new modified firefly algorithm. In: 3rd International conference reliability, infocom technologies and optimization (ICRITO) (Trends and Future Directions), IEEE
Grachten M, Arcos JL, de Mantaras RL (2014) Evolutionary optimization of music performance annotation. In: CMMR, 1–12
Hamadneh N, Sathasivam S, Tilahun SL, Choon OH (2012) Learning logic programming in radial basis function network via genetic algorithm. J Appl Sci (Faisalabad) 12(9):840–847
Hassanzadeh T, Kanan HR (2014) Fuzzy FA: a modified firefly algorithm. Appl Artif Intell 28:47–65
Hernandez S, Fontan A (2014) Cost optimization in bridge construction: application to launched bridges. Struct Congr 2014:2801–2812
Hongwei Z, Liwei T, Dongzheng W (2015) Research on improved firefly optimization algorithm based on cooperative for clustering. Int J Smart Home 9(3):205–214
Husselmann AV, Hawick KA (2011) Parallel parametric optimisation with firefly algorithms on graphical processing units, Technical Report CSTN-141
Jamil M, Yang X-S (2013) A literature survey of benchmark functions for global optimization problems. Int J Math Model Numer Optim 4(2):150–194
Jansi S, Subashini P (2015) A novel fuzzy clustering based modified firefly algorithm with chaotic map for mri brain tissue segmentation. MAGNT Res Rep 3(1):52–58
Kanimozhi T, Latha K (2013) An adaptive approach for content based image retrieval using Gaussian firefly algorithm. In: Huang DS et al. (eds) ICIC 2013, CCIS 375, pp 213–218
Kavousi-Fard A, Samet H, Marzbani F (2014) A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting. Expert Syst Appl 41:6047–6056
Kazemzadeh-Parsi MJ (2014) A modified firefly algorithm for engineering design optimization problems. IJST Trans Mech Eng 38(M2):403–421
Kazemzadeh-Parsi MJ (2015) Optimal shape design for heat conduction using smoothed fixed grid finite element method and modified firefly algorithm. IJST Trans Mech Eng 39(M2):367–387
Kazemzadeh-Parsi MJ, Daneshmand F, Ahmadfard MA, Adamowski J (2015) Optimal Remediation Design of Unconfined Contaminated Aquifers Based on the Finite Element Method and a Modified Firefly Algorithm. Water Resour Manage. doi:10.1007/s11269-015-0976-0
Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks IV, Nov 27–Dec 1, Perth, Australia, IEEE, 4, 1942–1948
Khan WA, Hamadneh NN, Tilahun SL, Ngnotchouye JMT (2016) A review and comparative study of firefly algorithm and its modified versions. In: Chapter 13 of optimization algorithms- methods and applications, associate Prof. Ozgur Baskan (Ed.), InTech, doi:10.5772/62472
Kwiecien J, Filipowicz B (2012) Firefly algorithm in optimization of queueing systems. Bull Pol Acad Sci Tech Sci 60(2):363–368
Lin X, Zhong Y, Zhang H (2013) An enhanced firefly algorithm for function optimisation problems. Int J Modell Identif Control 18(2):166–173
Liu C, Zhao Y, Gao F, Liu L (2015) Three-dimensional path planning method for autonomous underwater vehicle based on modified firefly algorithm. Math Probl Eng 2015, Article ID 561394, 10 pages
Long NC, Meesad P, Unger H (2015) A highly accurate firefly based algorithm for heart disease prediction. Expert Syst Appl 42:8221–8231
Lucia A, Xu J (1990) Chemical process optimization using Newton-like methods. Comput Chrm Eng 14(2):119–138
Lukasik S, Zak S (2009) Firefly algorithm for continuous constrained optimization task, ICCCI 2009. In: Ngugen NT, Kowalczyk R, Chen SM (eds) Lecture notes in artificial intelligence, 5796, 97–100
Maidl G, Schwerz de Lucena D, dos S Coelho L (2013) Economic dispatch optimization of thermal units based on a modified firefly algorithm. In: 22nd International congress of mechanical engineering (COBEM 2013), November. ABCM, RibeirÃčo Preto, SP, Brazil, pp 3–7
Manoharan GV, Shanmugalakshmi R (2015) Multi-objective firefly algorithm for multi-class gene selection. Ind J Sci Technol 8(1):27–34
Meena S, Chitra K (2015) Modified approach of firefly algorithm for non-minimum phase systems. Indian J Sci Technol 8(23):1–8

Mohammadi S, Mozafari B, Solimani S, Niknam T (2013) An adaptive modified firefly optimisation algorithm based on Hong's point estimate method to optimal operation management in a microgrid with consideration of uncertainties. Energy 51:339–348

Molga M, Smutnicki C (2016) Test functions for optimization needs, 2005, Retrieved Feb 2016. http://www.bioinformaticslaboratory.nl/twikidata/pub/Education/NBICResearchSchool/Optimization/VanKampen/BackgroundInformation/TestFunctions-Optimization.pdf

Negnevitsky M (2005) Artifcial intelligence: a guide to intelligent system. Henry Ling Limited, Harlow

Olamaei J, Moradi M, Kaboodi T (2013) A new adaptive modified firefly algorithm to solve optimal capacitor placement problem. In: 18th Electric power disteibution network conference, art. No. 6565962

Ondrisek B (2009) E-voting system security optimization. In: Proceedings of the 42nd Hawaii international conference on system sciences, Jan. 2009, 1–8

Othman MM, Hegazy YG, Abdelaziz AY (2015) A modified firefly algorithm for optimal sizing and siting of voltage controlled distributed generators in distribution networks. Period Polytech Electr Eng Comput Sci 59(3):104–109

Pan F, Ye C, Wang K, Jiangbo Cao (2013) Research on the vehicle routing problem with time windows using firefly algorithm. J Comput 8(9):2256–2261

Pike J, Bogich T, Elwood S, Finnoff DC, Daszak P (2014) Economic optimization of a global strategy to address the pandemic threat. Proc Natl Acad Sci 111(52):18519–18523

Poursalehi N, Zolfaghari A, Minuchehr A, Moghaddam HK (2013) Continuous firefly algorithm applied to PWR core pattern enhancement. Nucl Eng Des 258:107–115

Reddy PDP, Sekhar JNC (2014) Application of firefly algorithm for combined economic load and emission dispatch. Int J Rec Innov Trends Comput Commun 2(8):2448–2452

Ropponen A, Ritala R, Pistikopoulos EN (2010) Broke management optimization in design of paper production systems. In: Computer aided chemical engineering (20th European symposium on computer aided process engineering), 28, 865–870

Sahoo A, Chandra S (2013) Levy-flight firefly algorithm based active contour model for medical image segmentation, Contemporary Computing (IC3). In: Sixth international conference, IEEE, 159–162

Selvarasu R, Kalavathi MS (2015) TCSC placement for loss minimization using self adaptive firefly algorithm. J Eng Sci Technol 10(3):291–306

Selvarasu R, Kalavathi MS, Rajan CCA (2013) SVC placement for voltage constrained loss minimization using self-adaptive Firefly algorithm. Arch Electr Eng 62(4):649–661

Shafaati M, Mojallali H (2012) Modified firefly optimization for IIR system identification. Control Eng Appl Inf 14(4):59–69

Shakarami MR, Sedaghati R (2014) A new approach for network reconfiguration problem in order to deviation bus voltage minimization with regard to probabilistic load model and DGs. Int J Electr Comput Energ Electr Commun Eng 8(2):430–435

Subotic M, Tuba M, Stanarevic N (2012) Parallelization of the firefly algorithm for unconstrained optimization problems. Latest Adv Inf Sci Appl 264–269, ISBN: 978-1-61804-092-3

Subramanian R, Thanushkodi K (2013) An efficient firefly algorithm to solve economic dispatch problems. Int J Soft Comput Eng (IJSCE) 2(1):52–55

Sulaiman MH, Daniyal H, Mustafa MW (2012) Modified firefly algorithm in solving economic dispatch problems with practical constraints. In: IEEE international conference on power and energy (PECon), 2–5 December 2012, Kota Kinabalu Sabah, Malaysia

Sweitzer BJ (2008) Preoperative screening, evaluation, and optimization of the patient's medical status before outpatient surgery. Curr Opin Anaesthesiol 21(6):711–718

Tian Y, Gao W, Yan S (2012) An improved inertia weight firefly optimization algorithm and application. In: 2012 International conference on control engineering and communication technology. IEEE 64–68

Tilahun SL, Asfaw A (2012) Modeling the expansion of Prosopis Juliflora and determining its optimum utilization rate to control its invasion in afar regional state of ethiopia. Int J Appl Math Res 1(4):726–743

Tilahun SL, Ngnotchouye JMT (2016) Prey predator algorithm with adaptive step length. Int J Bio-Inspir Comput 8(4):195–204

Tilahun SL, Ngnotchouye JMT (2017) Firefly algorithm for discrete optimization problems: a survey. KSCE J Civil Eng 21(2):535–545

Tilahun SL, Ong HC (2012a) Bus timetabling as a fuzzy multiobjective optimization problem using preference based genetic algorithm. PROMET—traffic & transportation 24(3):183–191

Tilahun SL, Ong HC (2012b) Fuzzy preference of multiple decision makers in solving multiobjective optimization problems using genetic algorithm. Maejo Int J Sci Technol 6(02):224–237

Tilahun SL, Ong HC (2012c) Modified firefly algorithm. J Appl Math, Article ID 467631, 12 pages

Tilahun SL, Ong HC (2013) Vector optimisation using fuzzy preference in evolutionary strategy based firefly algorithm. Int J Op Res 16(1):81–95

Tilahun SL, Ong HC (2014) Prey-predator algorithm: a new metaheuristic optimization algorithm. Int J Inf Technol Decis Mak 13:1–22
Tilahun SL, Kassa SM, Ong HC (2012) A new algorithm for multilevel optimization problems using evolutionary strategy, inspired by natural adaptation. In: Anthony A, Ishizuka M, Lukose D (eds) PRICAI 2012, LNAI 7458. Springer, Berlin, pp 577–588
Tilahun SL, Hamadneh NN, Sathasivam S, Ong HC (2013) Prey-predator algorithm as a new optimization technique using in radial basis function neural networks. Res J Appl Sci 8(7):383–387
Tilahun SL, Ong HC, Ngnotchouye JM (2016) Extended prey predator algorithm with a group hunting scenario. Advances in Operations Research. doi:10.1155/2015/587103
Tilahun SL (2017) Prey predator hyperheuristic. Appl. Soft Comput 59:104–114
Verma OP, Aggarwal D, Patodi T (2016) Opposition and dimensional based modiïňĄed firefly algorithm. Expert Syst Appl 44:168–176
Villegas JG (2016) Using nonparametric test to compare the performance of metaheuristics. https://juangvillegas.les.wordpress.com/2011/08/friedman-test24062011.pdf. Retrieved Feb 2016
Volpato G, Maria E, Michielin Z, Ferreira SRS, Petrus JCC (2008) Optimization of the chicken breast cooking process. J Food Eng 84(4):576–581
Wadhwa Y, Kaur P, Kaur B (2014) Golomb Ruler sequence generation and optimization using modified firefly algorithm. SSRG Int J Electr Commun Eng (SSRG-IJECE) 1(5):1–8
Wang G, Guo L, Duan H, Liu L, Wang H (2012) A modified firefly algorithm for UCAV path planning. Int J Hybrid Inf Technol 5(3):123–144
Wang G-G, Guo L, Duan H, Wang H (2014a) A new improved FireïňĆy algorithm for global numerical optimization. J Comput Theor Nanosci 11:477–485
Wang B, Li D-X, Jiang J-P, Liao Y-H (2014b) A modified firefly algorithm based on light intensity difference. J Comb Optim. 31:1045–1060. doi:10.1007/s10878-014-9809-y
Yan X, Zhu Y, Wu J, Chen H (2012) An improved FireïňĆy algorithm with adaptive strategies. Adv Sci Lett 16:249–254
Yang X-S (2008) Nature-inspired metaheuristic algorithm, 2nd edn. Luniver Press, England
Yang XS (2010) Firefly algorithm, levy flights and global optimization. In: Bramer M, Ellis R, Petridis M (eds) Research and development in intelligent systems XXVI. Springer, London, pp 209–218
Yang X-S (2011) Review of metaheuristics and generalized evolutionary walk algorithm. Int J Bio-Inspir Comput 3(2):77–84
Yang X-S (2013) Multiobjective ïňĄreïňĆy algorithm for continuous optimization. Eng Comput 29:175–184
Yu S, Yang S, Su S (2013) Self-adaptive step firefly algorithm. J Appl Math 832718:8
Yu S, Zhu S, Ma Y, Mao D (2015a) A variable step size ïňĄreïňĆy algorithm for numerical optimization. Appl Math Comput 263:214–220
Yu S, Mao D, Zhu S, Ma Y (2015b) Enhancing firefly algorithm using generalized opposition-based learning. Computing 97:741–754
Yu S, Su S, Huang L (2015) A simple diversity guided firefly algorithm. Kybernetes 44(1):43–56