Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

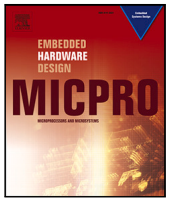
Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.

Microprocessors and Microsystems 98 (2023) 104778

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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



A novel firefly algorithm approach for efficient feature selection with COVID-19 dataset

Nebojsa Bacanin [a,*], K. Venkatachalam [b], Timea Bezdan [a], Miodrag Zivkovic [a], Mohamed Abouhawwash [c,d]

[a] Singidunum University, Danijelova 32, 11000 Belgrade, Serbia
[b] Department of Applied Cybernetics, Faculty of Science, University of Hradec Králové, 50003 Hradec Králové, Czech Republic
[c] Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt
[d] Department of Computational Mathematics, Science, and Engineering (CMSE), Michigan State University, East Lansing, MI 48824, USA

ARTICLE INFO

Keywords:
Firefly algorithm
Swarm intelligence
Quasi-reflection-based learning
Feature selection
Genetic operators
COVID-19 dataset

ABSTRACT

Feature selection is one of the most important challenges in machine learning and data science. This process is usually performed in the data preprocessing phase, where the data is transformed to a proper format for further operations by machine learning algorithm. Many real-world datasets are highly dimensional with many irrelevant, even redundant features. These kinds of features do not improve classification accuracy and can even shrink down performance of a classifier. The goal of feature selection is to find optimal (or sub-optimal) subset of features that contain relevant information about the dataset from which machine learning algorithms can derive useful conclusions. In this manuscript, a novel version of firefly algorithm (FA) is proposed and adapted for feature selection challenge. Proposed method significantly improves performance of the basic FA, and also outperforms other state-of-the-art metaheuristics for both, benchmark bound-constrained and practical feature selection tasks. Method was first validated on standard unconstrained benchmarks and later it was applied for feature selection by using 21 standard University of California, Irvine (UCL) datasets. Moreover, presented approach was also tested for relatively novel COVID-19 dataset for predicting patients health, and one microcontroller microarray dataset. Results obtained in all practical simulations attest robustness and efficiency of proposed algorithm in terms of convergence, solutions' quality and classification accuracy. More precisely, the proposed approach obtained the best classification accuracy on 13 out of 21 total datasets, significantly outperforming other competitor methods.

1. Introduction

Machine learning has recently become very important and one of the most prominent research fields because it can be applied to various domains, e.g. image and speech recognition, self-driving cars, traffic and stock market predictions, product recommendations, medical diagnosis, spam email filtering, etc. In general, all machine learning tasks refer to either classification or regression.

Besides machine learning, the concept of the internet of things (IoT) has also recently emerged. According to one of the basic definitions, the IoT refers to the next generation of engineered systems which are comprised of connected devices that are able to collect, process, and share large amounts of data [1]. Moreover, the IoT concept has been extended with the new paradigm — brain-empowered IoT or cognitive IoT (CIoT) [2]. This paradigm broadens the capabilities of existing IoT systems and incorporates intelligence in them by using cognitive computing.

However, the domain of IoT is confronting many challenges. To efficiently and effectively run a complex network of interconnected devices, challenges like link failures, coverage, scalability, security, and interoperability may arise. Moreover, in the era of IoT and CIoT in order to find patterns and derive conclusions from huge datasets that are collected, artificial intelligence technologies such as machine learning are needed. Thus, the machine learning models and algorithms are very important aspects of IoT and CIoT in a way that they can be used for improving the IoT and CIoT quality of service (QoS), as well as for making intelligence decisions from massive datasets. It is said that machine learning plays a vital role in driving the evolution of these technologies [3,4].

* Corresponding author.
E-mail addresses: nbacanin@singidunum.ac.rs (N. Bacanin), venkatachalam.k@ieee.org (K. Venkatachalam), tbezdan@singidunum.ac.rs (T. Bezdan), mzivkvoic@singidunum.ac.rs (M. Zivkovic), abouhaww@msu.edu (M. Abouhawwash).

https://doi.org/10.1016/j.micpro.2023.104778
Received 7 July 2021; Received in revised form 12 May 2022; Accepted 29 January 2023
Available online 6 February 2023
0141-9331/© 2023 Elsevier B.V. All rights reserved.

Like any other area, machine learning is facing many challenges and open issues. From the data preprocessing perspective one of the most important challenges is feature selection. Real-world datasets, such are those that are acquired by IoT/CIoT systems, often include many irrelevant and redundant features, which can even degrade the performance of a classifier due to the relatively large search domain. For that reason, especially in the IoT/CIoT context, to shorten the training time and to enable real-time decision-making, feature selection is of profound importance.

Feature selection belongs to the group of NP-hard combinatorial optimization problems because of a large search space, e.g. in datasets with N features, 2^N possible feature combinations exist. Many approaches were devised for tackling this challenge [5,6] however, methods like exhaustive search are practically inapplicable and traditional algorithms such is FOCUS are not able to generate satisfying results. It is proven that when tackling NP-hard challenges such is feature selection, the most promising results can be achieved by using metaheuristics-based approaches [7,8].

By surveying available literature, it can be seen that many metaheuristics algorithms, especially from the group of nature-inspired approaches (swarm intelligence and evolutionary algorithms — EA), were successfully adapted for solving feature selection [9], but it was also observed that there is still capacity for improvements in terms of classification accuracy, training time and the number of utilized features. One reason for that is that many of the proposed metaheuristics suffer from stagnation in sub-optimal domains [10,11].

The research question addressed in this paper can be formulated as follows: Is it possible to obtain better classification accuracy and/or reduce the number of features by utilizing FA for feature selection in comparison to other available contemporary methods? Consequently, the, [12,13] major goal of the research proposed in this manuscript is to address the feature selection challenge further by achieving better classifications accuracy and/or to utilize a smaller number of features than other existing traditional and metaheuristics-based methods. From the literature survey, it also can be concluded that the potential of one of the most efficient swarm intelligence approaches, the firefly algorithm (FA), was not fully investigated for feature selection. This motivated authors to devise a novel FA method and to adapt it for tackling this issue.

Contributions of the proposed research can be summed as follows: proposing a new FA approach that is able to efficiently tackle feature selection in terms of classification accuracy, and computational time and overcoming deficiencies of original FA and outscoring performance of other enhanced FA methods proposed in the modern literature by using genetic operators and quasi-reflexive-based learning.

It is noted that the proposed research represents an extension of the previous investigation with FA adaptations for feature selection [14]. In [14,15] one more enhanced FA is shown, however method proposed in this paper by a large amount outperforms the former FA version. For the purpose of this study, the original FA and method proposed in [14,16] are also implemented.

Following the most commonly used practice from modern computer science and also to validate research premises and contributions, novel FA was firstly tested against the set of 18 standard unconstrained benchmarks and compared with original FA and 4 other enhanced state-of-the-art FA implementations. Afterward, practical simulations for feature selection with 21 well-known UCL , 1 COVID-19, and 1 microcontroller dataset were conducted and comparative analysis with other state-of-the-art methods was performed.

The remainder of the manuscript is structured as follows. In Section 2, basic theoretical background from machine learning, feature selection and swarm intelligence is given along with the review of relevant literature. Proposed novel FA approach is described in Section 3. Practical simulations with comparative analysis and discussion are given in Section 4, while Section 5 provides final remarks and future research perspectives from this very important domain.

2. Background and relevant literature survey

Machine learning is very promising field from the area of artificial intelligence. In most basic terms, machine learning is the ability of a computer system to learn without being explicitly programmed. At even the most fundamental, machine learning employs pre-programmed algorithms that collect and analyze input data in order to anticipate output values within an acceptable range. As incoming data is fed into these algorithms, they learn and optimize their processes to enhance performance, resulting in 'intelligence' development throughout time. Machine learning algorithms can be categorized as supervised, unsupervised, semi-supervised and reinforcement. In the case of supervised learning, where a labeled data is used for training the model, two most common tasks are classification and regression.

As it was already noted in Section 1, one of the most important challenges from machine learning domain is feature selection. The goal of this task is to select subset of complementary features from available pool of characteristics to establish better classification accuracy. Here, the attribute 'complementary' plays an important role because there may be two, or multi-way interactions among features [10]. For example, individual relevant features may become irrelevant when working with others. Similarly, individual redundant and not important features may become significant when paired with others [17]. This problem becomes exponentially harder for tackling when the number of features is increasing. Moreover, feature selection challenge often encompasses two contradictory objectives: maximizing classification accuracy and minimizing the number of employed features.

Feature selection is a pre-processing method in data mining and machine learning, that reduces the dimensionality of data by removing the noise and irrelevant attributes and hopefully results in optimal or near-optimal feature subset. Two key concepts in feature selection are the criteria of evaluation and the strategy of selection. Considering the criteria of evaluation, three main types of feature selection approaches exist wrapper-based method, filter-based method, and embedded methods. The filter-based methods, such as Gini Index, Information Gain, Relief, FOCUS, and Chi-Square [5] use statistical measures to assign scores to each feature, and based on the scores, it makes the ranking of features and chooses a subset. The wrapper-based method uses machine learning techniques to select the optimal attribute subset. Most commonly, regardless of higher computational costs, the wrapper-based methods are employed in feature selection processes, because these methods result in better classification accuracy. The embedded methods are mixed with the filter and wrapper methods, and as such, they take the advantage of wrapper-based and filter-based approaches.

Due to the large search space, the most promising methods for tackling feature selection are metaheuristics, especially nature-inspired such as EA and swarm intelligence. Metaheuristics are often used as wrapper methods for feature selection challenges. For example, the genetic algorithm (GA) proved an efficient wrapper-based method for this task [18–20].

Swarm intelligence is a very efficient and robust optimizer for many practical NP-hard problems from various domains [21–23]. These methods emulate natural systems by performing exploitation and exploration processes. The basic idea behind this group of algorithms is to apply and adopt guided random search mechanisms from nature into the optimization process. Some of the most notable examples of swarm intelligence include ant colony optimization (ACO) [24], particle swarm optimization (PSO) [25], artificial bee colony (ABC) [26], bat algorithm (BA) [27] and the FA [28]. Recently, many swarm intelligence approaches were applied for feature selection [29–31] and a detailed survey can be seen in [9]. Several recent publications in cutting-edge journals also deal with swarm intelligence methods to reduce the number of features. Research published in [32] proposes a hybrid brainstorm optimization method for feature selection, with encouraging results. Another promising research by Zivkovic et al. [33]

utilized an improved salp swarm algorithm to tackle this issue. Modified binary ant lion algorithm was used in the paper by Strumberger et al. [34], while [35] applied the hybridized sine cosine algorithm for the same task. Kareem et al. [36] suggested the grasshopper algorithm for feature selection, and obtained excellent results on the intrusion detection datasets. It is also worth mentioning that swarm intelligence has also been adopted for other machine learning challenges, specifically hyperparameters' optimization [22,37], artificial neural network (ANN) training [38], as well as in for many others [39].

3. Proposed novel FA approach

The FA metaheuristics, introduced by Yang [28], is motivated by the flashing and social characteristics of fireflies. Since the real-world, natural system is relatively complex and sophisticated, the FA models it by using several approximation rules [28].

Brightness and attractiveness of fireflies is used for modeling fitness function in a way that attractiveness in most typical FA's implementations depends on the brightness, that is in turn determined by the objective function value. In the case of minimization problems, it is formulated as [28]:

$$I(x) = \begin{cases} \dfrac{1}{f(x)} & \text{, if } f(x) > 0 \\ 1 + |f(x)| & \text{, otherwise} \end{cases} \tag{1}$$

where $I(x)$ represents attractiveness, and $f(x)$ denotes the value of objective function at location x.

Light intensity, hence the attractiveness of individual decrease, as the distance from the light source increase [28]:

$$I(r) = \frac{I_0}{1 + \gamma r^2} \tag{2}$$

where $I(r)$ represents light intensity at distance r, while I_0 stands for the light intensity at the source. Furthermore, for modeling real natural systems, where the light is partly absorbed by its surroundings, the FA makes use of the γ parameter, which represents the light absorption coefficient. In most FA's versions, the combined effect of inverse square law for distance and the γ coefficient is approximated with the following Gaussian form [28]:

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

Moreover, each firefly individual also utilizes attractiveness β, which is directly proportional to the light intensity of a given firefly and also depends on the distance, as it is shown in Eq. (4).

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

where parameter β_0 designates attractiveness at distance $r = 0$. It should be noted that in practice, Eq. (4) is often replaced by Eq. (5) [28]:

$$\beta(r) = \frac{\beta_0}{1 + \gamma r^2} \tag{5}$$

Based on the stated above, the basic FA's search equation for a random individual i, that moves in iteration $t + 1$ to a new location x_i towards individual j with greater fitness is given as [28]:

$$x_i^{t+1} = x_i^t + \beta_0 \cdot e^{-\gamma r_{i,j}^2}(x_j^t - x_i^t) + \alpha^t(\kappa - 0.5) \tag{6}$$

where α stands for the randomization parameter, random number drawn from Gaussian or uniform distribution is denoted as κ and $r_{i,j}$ represents the distance between two observed fireflies i and j. Typical values that establish satisfying results for most problems for β_0 and α are 1 and $[0, 1]$, respectively.

The $r_{i,j}$ is Cartesian distance, which is calculated by using Eq. (7).

$$r_{i,j} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{D}(x_{i,k} - x_{j,k})^2} \tag{7}$$

where D marks the number specific problem parameters.

3.1. Original FA's deficiencies

Due to the fact that the FA exhibits satisfying performance for many benchmark [40] and practical challenges [41], it has been thoroughly tested from both, theoretical and practical perspective by the researchers. However, findings of some previous research suggest that the basic FA manifest some deficiencies that can be summarized as the lack of exploration and inadequate exploitation–exploration trade-off [42,43].

Due to the lack of diversification power, in some runs, the algorithm converges to sub-optimal parts of the search space and performs exploitation within this region. This leads to the poor solutions quality at the end of a run, and eventually to the worse mean values for a batch of runs. Moreover, it can be noticed that the basic FA's search equation Eq. (6) does not include clearly emphasized diversification component. Only the randomization parameter α is used to control diversification and balance with the exploitation, which is not enough.

Some researchers tried to address this issue by including dynamic α parameter, that is at the beginning of a run larger (empowered exploration), and afterwards it is being gradually decreasing with the progress of iterations, moving gradually balance from exploration towards exploitation [44]. However, based on the simulations conducted for the purpose of this manuscript, it is observed that this behavior can only slightly eliminated this issue and as such it does not represent the final solution.

3.2. Proposed improvements

To address issues of basic FA metaheuristics, novel FA approach proposed in this manuscript incorporates the following in its basic version:

- genetic operators (GO) - uniform crossover and Gaussian mutation and
- quasi-reflection-based learning mechanism (QRBL).

Motivated by introduced upgrades, proposed approach is referenced as the genetic operators quasi-reflected FA (GOQRFA). Besides addressing the absence of explicit diversification and the inappropriate balance between intensification and diversification, by introducing proposed changes, exploitation process of the original FA is also further improved.

By applying GO in early iterations by recombining solutions from novel regions of the search space, the search process is less likely to be trapped in sub-optimal domains and more efficient exploration is performed. On contrarily, in later iterations, GO enable fine-tuning around the domain where an optimum resides and final solutions' quality is improved.

Implications of QRBL to the algorithm robustness are also two-fold. Based on the findings of previous research, solutions diversity in early iterations, as well as the convergence speed in later phases of execution, can be dramatically boosted if QRBL mechanism is applied [45].

Besides, the GOQRFA also incorporates dynamically adjusted step size parameter α as suggested in [44].

In the context of GO, potential solutions is encoded as chromosome that consists of genes, where each gene represents one parameter of objective function. In the proposed approach, uniform crossover and Gaussian mutation operators are applied on the gene level with gene probability — GP. In the case of uniform crossover, each gene will be exchanged between two parent solutions with probability p. If the p is closer to 0.5, then the gene exchange will be more frequent and exploratory (global) behavior will be exhibited. Conversely, when the p is closer to 0 or 1, uniform crossover will be more locally oriented and exploitation around the current solutions will be emphasized.

In GOQRFA, when uniform crossover is applied, for each pair of parent solutions, only one offspring is created. As an example, when

recombining current best solution x_{best} and one random solution x_{rnd}, for each parameter j, offspring x_{off} is generated in the following manner:

$$x_{off,j} = \begin{cases} x_{best,j} & \text{, if } \phi \leq p \\ x_{rnd,j} & \text{, otherwise} \end{cases} \tag{8}$$

where ϕ is a pseudo-random number drown from the uniform distribution.

When an offspring solution is generated, it is being subdued to mutation. Among various types of mutations used in methods from the modern literature, uniform, polynomial, and Gaussian are considered to be classical ones [46]. Since the Gaussian mutation operator proved to be very efficient for tackling NP-hard problems by preventing the loss of diversity during the search process [46], this operator is utilized.

Introduced by Bäck and Schwefel, Gaussian mutation is founded upon the Gaussian density function [47]:

$$f_{gaussian(0,\sigma^2)}(\theta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\theta^2}{2\sigma^2}} \tag{9}$$

where the variance of candidate solutions is denoted as σ^2.

Gaussian density function can be reduced for the mean of 0 and standard deviation σ of 1. In the context of GOQRFA, Gaussian distributed random vector $G(\theta)$ can be generated and applied for each parameter j of solution h_{off}:

$$x_{off,j} = x_{off,j} \cdot (1 + G(\theta_j)) \tag{10}$$

Basic assumption that served as an inspiration for incorporating QRBL mechanism in the FA is that when creating new solutions by using quasi-opposite numbers, there is a higher probability that such novel solutions will be close to optimum region, than simply sampling random individuals from the search space. The QRBL [48] originates from the opposition-based learning (OBL) and quasi-opposition-based learning (QOBL) and it was proven that the QRBL is more efficient in performing both exploitation and exploration than other two mechanism [48].

The quasi-reflected component j of solution x (x_j^{qr}) can be calculated as:

$$x_j^{qr} = \text{rnd}\left(\frac{lb_j + ub_j}{2}, x_j\right) \tag{11}$$

where lb_j and ub_j represent lower and upper bound of jth component, respectively, $\frac{lb_j + ub_j}{2}$ indices arithmetic mean (center) of the interval $[lb_j, ub_j]$, while $\text{rnd}\left(\frac{lb_j + ub_j}{2}, x_j\right)$ generates uniformly distributed pseudo-random from the interval $\left[\frac{lb_j + ub_j}{2}, x_j\right]$.

Finally, as the third upgrade of original FA, the GOQRFA incorporates dynamic step size α with a greater value at the beginning of a run, hence the more emphasized global search, which is further being decreased moving balance towards exploitation. The random parameter α starts from its initial value α_0 until it reaches minimum threshold α_{min} as iterations advance:

$$\alpha^{t+1} = \alpha^t \cdot \left(1 - \frac{t}{MaxIter}\right), \tag{12}$$

where t and $t+1$ denote current and next iterations, respectively, while the $MaxIter$ is the maximum iteration number in one run of an algorithm.

3.3. The GOQRFA implementation

As noted in Section 3.2, the GOQRFA addresses insufficient exploration and inadequate intensification–diversification balance of the original FA by using GO and QRBL mechanism. According to practical simulations, it is shown that by using both procedures, state-of-the-art results can be obtained.

Since the exploration in early, while the exploitation in later iterations should be amplified, two different crossover mechanisms were employed. With this purpose, additional control parameters were included in the GOQRFA: the number of replaced solutions nrs, which is used for adjusting the number of worst solutions from the population that are replaced with offspring solutions, and exploration break point (ebp) that controls which of the above-mentioned crossover mechanisms will be triggered.

In the first ebp iterations, the diversification uniform crossover (DUC) mechanism is triggered as follows: the nrs worst solutions from the population are replaced with offspring individuals generated by performing recombination between the completely random solution (x_{rnd}) from the search domain and randomly chosen existing solutions from the population (x_{prnd}) by using Eq. (8) with probability p on the gene level. A completely random solution is created in the same way as in the initialization phase (Eq. (14)). The DUC mechanism is executed in each iteration.

After ebp, with the goal of improving exploitation around the current best solutions, intensification uniform crossover (IUC) mechanism is triggered as follows: nrs worst solutions from the population are replaced with the offspring solutions generated by performing recombination between the first best (x_{best1}) and the second-best (x_{best2}) solutions from the population by utilizing Eq. (8) with uniform crossover probability p on the gene level. If $nrs > 1$, then for each replaced solution, the new hybrid is generated. By conducting extensive empirical simulations, it was observed that if the IUC is triggered too frequently, the population may lose diversity and may converge to sub-optimal solutions. As the iterations progress, the IUC frequency should increase because the algorithm is progressing towards the optimum region of the search space. To control this behavior, in each iteration, the IUC is executed with probability $IUCp$ (IUC probability), which is increasing directly proportionally to the current iteration from the initial value $IUCp_0$:

$$IUCp^{t+1} = IUCp^t \cdot \left(1 + \frac{t}{MaxIter}\right) \tag{13}$$

In both cases (IUC and DUC) Gaussian mutation on the gene level is executed for each offspring solution by applying Eq. (10). However, different mutation probabilities are applied for each gene (solution's parameter) - mutation probability for diversification (mpd) and mutation probability for intensification (mpi) for offspring generated during DUC and IUC phases, respectively. Since stronger exploration is required in the first ebp iterations, the mpd is higher than mpi. These values, which are empirically determined depend on the number of solutions parameters' D and they are calculated as: $mpd = \frac{1}{D}$ and $mpi = \frac{1}{2D}$.

The QRBL strategy is used along with the GO. This procedure is included in the initialization phases, along with the solutions' update phase. In the initialization phase, each solution parameter j for the every solution x_i from the initial population ($P_{init} = \{X_{i,j}\}, i = 1, 2, 3..., NS; j = 1, 2, \ldots, D$) is generated by using standard initialization expression [49]:

$$x_{i,j} = lb_j + (ub_j - lb_j) \cdot rand \tag{14}$$

where $rand$ is uniformly distributed random number from the interval $[0, 1]$.

Then, the QRBL is applied (Eq. (11)) to determine quasi-reflective solution of each individual from the population, and quasi-reflective initial population ($P_{init}^{qr} = \{X_{i,j}^{qr}\}, i = 1, 2, 3..., NS; j = 1, 2, \ldots, D$) is generated. Both populations are then merged together ($P_{init} \cup P_{init}^{qr}$) and sorted in descending order according to fitness value and NS best solutions are selected as the new initial population.

Similarly, during the update phase, in each iteration, quasi-reflective population P^{qr} of updated population P is created. Also, like in the initialization phase, populations are then merged ($P \cup P^{qr}$) and

Table 1
GOQRFA control parameters summary.

Parameter description	Notation	Type
Number of solutions in population	NS	FA standard (static)
Maximum iteration number	$MaxIter$	FA standard (static)
Absorption coefficient	γ	FA standard (static)
Attractiveness parameter at $r = 0$	β_0	FA standard (static)
Randomization (step) parameter	α	FA standard (dynamic)[a]
Initial value of step parameter	α_0	FA standard (static)
Minimum value of step parameter	α_{min}	FA standard (static)
Exploration break point	ebp	GOQRFA specific (static)
Number of replaced solutions	nrs	GOQRFA specific (static)
Uniform crossover probability	p	GOQRFA specific (static)
Intensification uniform crossover probability	$IUCp$	GOQRFA specific (dynamic)[b]
Initial value of intensification uniform crossover probability	$IUCP_0$	GOQRFA specific (static)
Mutation probability for diversification	mpd	GOQRFA specific (fixed)
Mutation probability for intensification	mpi	GOQRFA specific (fixed)

[a]Changes according to Eq. (12).
[b]Changes according to Eq. (13).

sorted in descending order according to fitness value, and NS best solutions are selected to propagate to the next iteration.

The GOQRFA employs the same basic search procedure as in the original FA metaheuristics (Eq. (6)).

In summary, the GOARFA utilizes both static and dynamic parameters, among some of them are inherited from the original FA, while others are GOARFA specific. All parameters are shown in Table 1.

Detailed pseudo-code of proposed GOQRFA metaheuristics is given in Algorithm 1.

Algorithm 1 The GOQRFA pseudo-code

```
Initialize main metaheuristics control parameters NS and MaxIter
Initialize search space parameters D, ub_j and lb_j
Initialize GOQRFA control parameters γ, β_0, α_0, α_min, nrs, ebp, p, IUC_p0, mpd and mpi
Generate initial random population P_init = {X_i,j}, i = 1,2,3...,NS; j = 1,2,...D using
Eq. (14) in the search space
Generate quasi-reflective initial population P_init^qr by using QRBL strategy
Intensity of light I_i (fitness) at position x_i is defined by f(x)
Create population P by merging populations P_init and P_init^qr, calculate fitness of all solutions,
sort individuals in descending order according to quality and select best NS individuals
while t < MaxIter do
  for i = 1 to NS do
    for k = 1 to i do
      if I_k < I_i then
        Move solution k in the direction of individual i in D dimensions (Eq. (6))
        Attractiveness changes with distance r as exp[−γr] (Eq. (4))
        Evaluate new solution, replace the worse individual with better one and update
        intensity of light (fitness)
      end if
    end for
  end for
  if t ≤ ebp then
    Execute DUC mechanism and replace nrs worst solutions with hybrid between x_rnd
    and x_prnd individuals using Eq. (8) and apply Gaussian mutation (Eq. (10)) with
    mpd probability
  else
    Execute IUC mechanism with probability IUC_p(t) and replace nrs worst solutions
    with hybrid between x_best1 and x_best2 individuals using Eq. (8) and apply Gaussian
    mutation (Eq. (10)) with mpi probability
  end if
  Generate quasi-reflective population P^qr by using QRBL strategy
  Merge populations P and P^qr, calculate fitness of all solutions, sort individuals in
  descending order according to quality and select best NS individuals
  Update dynamic parameters α and IUC_p for next iteration t+1 according to Eqs. (12)
  and (13), respectively
end while
Return the best individual x_i from the population
```

3.4. The GOQRFA complexity and limitations

The number of objective function evaluations is usually taken to calculate swarm intelligence algorithm complexity [40]. In the basic FA algorithm, fitness is calculated in the initialization phase and in the solutions' updating phase. In the updating phase, basic FA has one main loop for iterations t and two inner loops going through NS solutions [40].

Thus, including the initialization phase, the complexity in the worst case of basic FA metaheuristics is $O(NS) + O(NS^2t)$. However, if NS is relatively large, it is possible to use one inner loop by ranking the attractiveness or brightness of all fireflies using sorting algorithms, and in this case complexity is $O(NS) + O(NSt\log(NS))$ [40].

The complexity of the GOQRFA is higher than the original FA due to the application of the QRBL mechanism. The UCD and UCI mechanisms are not counted since in these phases, objective function evaluations are performed only nrs times in each iteration, and nrs is typically small (1 or 2). The QRBL is applied in the initialization phase and after the solution's update phase in each iteration, and in both cases, an additional NS number of function evaluations are performed. Therefore, the complexity of the proposed GOQRFA in the worst-case scenario can be expressed as: $2 \cdot O(NS) + O(NS^2t) + O(SNt)$.

Limitation of proposed GOQRFA is additional number of control parameters which makes relatively hard for the researcher to find a proper values of control parameters for a specific problem. This issue is similar to hyperparameters' optimization challenge from ML learning domain.

However, additional number of control parameters is justified because the GOQRFA shown substantial performance improvements over the original FA for benchmarks challenges and also for the practical feature selection problem in ML domain, as it is shown in Section 4.

4. Simulations and discussion

Following the good practice from modern computer science literature, practical (experimental) section of the manuscript is divided into two parts. In the first part, validation of proposed method for standard set of unconstrained benchmarks is shown along with comparison between proposed GOQRFA and other most recent improved FA implementations, which are presented in [50]. Afterwards, in the second part of this section, adaptation of GOQRFA for feature selection challenge along with simulation results on standard datasets and comparative analysis with most recent metaheuristics are presented.

To validate improvements of GOQRFA over the original FA, both metaheuristics are implemented and tested. Both implementations are created in Python by using core (built-in), as well as specific data science and machine learning Python libraries: numpy, scipy, pandas, scikitlearn, pyplot and seaborn.

All experiments were conducted on Intel® CoreTM i7-8700K CPU and 32 GB of RAM running under Windows 10 x64 operating system computer platform.

Table 2
Setup of GOQRFA control parameters.

Parameter and notation	Value
Absorption coefficient γ	1.0
Attractiveness parameter at $r = 0$ β_0	1.0
Randomization (step) parameter α	Eq. (12)
Initial value of step parameter α_0	0.5
Minimum value of step parameter α_{min}	0.1
Exploration break point ebp	$MaxIter/3$
Number of replaced solutions nrs	1
Uniform crossover probability p	0.5
Intensification uniform crossover probability $IUCp$	Eq. (13)
Initial value of intensification uniform crossover probability $IUCP_0$	0.2
Mutation probability for diversification mpd	$1/D$
Mutation probability for intensification mpi	$1/2 \cdot D$

4.1. Standard unconstrained benchmarks simulations

In this subsection, we first show benchmark functions details that were utilized in simulations along with GOQRFA control parameters', followed by comparative analysis with other FA versions, results' visualization, and discussion.

4.1.1. Benchmark details and GOQRFA settings

Since the GOQRFA metaheuristics employs more control parameters than the standard FA, extensive simulations on classical unconstrained benchmarks were conducted to determine the best parameter setup. The goal is to find control parameters' values that will on average for all test instances accomplish satisfying results. Unfortunately, the only way to find these values is by employing the 'trial and error' strategy, which is very time-consuming. However, based on authors' previously acquired domain experience with the FA [41,49], as well as other swarm algorithms [23], proper parameters' values were determined relatively easy.

The GOQRFA control parameter values that were used in experiments are summarized in Table 2.

It is noted that the parameter p is hard-coded in the GOQRFA and it is not adjustable. In this way, the adjusted balance between exploration and exploitation for a uniform crossover operator is maintained during the whole run of the algorithm. Also, the nrs is set to 1 regardless of the number of solutions in the population (NS) and maximum iterations ($MaxIter$). It was empirically discovered that if the value of this parameter is higher, then before ebp iterations, the exploration–exploitation balance will be exceedingly moved towards exploration, and after ebp iterations search process will be too much unbalanced towards exploitation.

Unconstrained benchmark instances details utilized in simulations are shown in Table 3. The same test-beds as in [50] were utilized to make a comparative analysis with improved FA approaches shown in [50] more realistic.

Test instances $f1, f3, f4, f5, f6, f7, f14$ and $f15$ are retrieved from the Congress on Evolutionary Computation (CEC) benchmark suite, while other functions are standard test instances utilized to evaluate algorithms' convergence and solutions' quality.

Functions shown in Table 3 have different characteristics. Benchmarks $f1, f2, f5, f7, f8, f12$ and $f14$ are complex unimodal functions with only global optimum and they are primarily used to test the convergence speed of an algorithm. However, other test functions $f3, f4, f6, f9, f10, f11, f13$ and $f15$ belong to the group of multimodal benchmarks with many local optima and they are used to test the algorithm's exploration ability to jump out of the local extreme value. Additionally, the proposed method is tested on highly complex two-dimensional functions $f16, f17$ and $f18$ with multiple local minima.

4.1.2. Comparative analysis and discussion

As noted above, in the comparative analysis original FA metaheuristics along with the following improved versions is included: dynamic adaptive weight firefly algorithm (WFA) [51], chaotic firefly algorithm based on logistic map (CLFA) [52], Levy flights FA (LFA) [53], variable step size firefly algorithm for numerical Optimization (VSSFA) [54] and dynamically adaptive firefly algorithm with global orientation (GDAFA) [50].

For the purpose of proposed research and to analyze and evaluate improvements of the GOQRFA over original FA, the basic FA version is implemented and tested, while test results for other improved FA's implementations were retrieved from [50]. In conducted experiments, basic FA with dynamic parameter $alpha$ (Eq. (12)), is used, and much better results than those that are reported in [50] were obtained.

Control parameters' of opponent metaheuristics included in the comparative analysis can be seen from [50].

Simulations with 10, 30 and 100-dimensional ($D = [10, 30, 100]$) benchmark instances $f1$–$f15$ shown in Table 3 were performed and compared best, mean (average) and worst metrics were generated in 50 independent runs.

Algorithms in [50] were tested with 20 solutions in the population (NS=20) and maximum number of 1,000 iterations per run ($MaxIter$=1000). However, since the GOQRFA employs the QRBL mechanism, its complexity in terms of number fitness function evaluations (FFE) is slightly larger than other algorithms. Due to this, the number of solutions in the population for GOQRFA simulations is decreased, while using the same number of iterations, as in [50].

The worst-case complexity in terms of FFEs of all opponent approaches is $O(SN)+O(SN^2t)$, and in the case of GOQRFA is $O(2SN)+O(SN^2t)+O(SNt)$. Mathematically was derived that with 1,000 iterations, the NS of GOQRFA should be set to 18 for fair comparative analysis.

Simulation results with 10, 30 and 100 dimensional problems are shown in Tables 4, 5 and 6, respectively. The obtained results for 2-dimensional function ($f16$–$f18$) are presented in the Table 7. If there is an algorithm that obtains the best results for some performance metric, such results are marked bold with a slightly larger font in all tables.

From Table 4, which shows simulation results with 10-dimensional instances, can be seen that the GOQRFA and GDAFA managed to find global optimum relatively easy and in almost all tests for all runs found the global best solution.

Similarly, as in the tests with 10-dimensional problems, GOQRFA and GDAFA proved to be the two best-improved FA metaheuristics in simulations with 30-dimensional problems (Table 5). The GOQRFA outscored GDAFA for all three metrics in $f13$ instance and obtained better worst and mean metrics in $f1$, $f2$, $f4$, $f5$, $f7$ and $f8$ instances. In all benchmark tests, except $f13$, both algorithms showed equal performance when comparing the best metrics. The GDAFA outperformed the proposed GOQRFA in the worst indicator for $f3$ test, and the mean indicator in the case of $f6$ and $f15$ instances. In the case of simulation for $f9$ benchmark, GDAFA obtained better

Table 3
Unconstrained benchmark function details.

ID	Name	Search range	Formulation	Optimum
f1	Sphere	$[-100, 100]^D$	min $f(x) = \sum_{i=1}^{D} x_i^2$	0
f2	Moved Axis Function	$[-5.12, 5.12]^D$	min $f(x) = \sum_{i=2}^{D} 5ix_i^2$	0
f3	Griewank	$[-100, 100]^D$	min $f(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1$	0
f4	Rastrigin	$[-5.12, 5.12]^D$	min $f(x) = 10n + \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i)]$	0
f5	The Schwefel's Problem 1.2	$[-100, 100]^D$	min $f(x) = \sum_{i=1}^{D} \sum_{j=1}^{i} x_j^2$	0
f6	Ackley	$[-32, 32]^D$	min $f(x) = -a \times exp(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{n} \cos(cx_i)) + a + exp(1)$, where $a = 20, b = 0.2$	0
f7	Powell Sum	$[-1, 1]^D$	min $f(x) = \sum_{i=1}^{D} \lvert x_i \rvert^{i+1}$	0
f8	Sum Squares	$[-10, 10]^D$	min $f(x) = \sum_{i=1}^{D} ix_i^2$	0
f9	Schwefel 2.22	$[-100, 100]^D$	min $f(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	0
f10	Powell Singular	$[-4, 5]^D$	min $f(x) = \sum_{i=1}^{D/4} [(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{xi})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} + x_{4i})^4]$	0
f11	Alpine	$[-10, 10]^D$	min $f(x) = \sum_{i=1}^{D} \left\lvert x_i \sin\left(x_i + 0.1x_i\right) \right\rvert$	0
f12	Inverse Cosine-Wave Function	$[-100, 100]^D$	min $f(x) = \sum_{i=1}^{D-1} (exp(-\frac{x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1}}{8}) \times \cos(4\sqrt{x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1}}))$	−D+1
f13	Pathological	$[-100, 100]^D$	min $f(x) = \sum_{i=1}^{D-1} \left(0.5 + \frac{\sin^2\left(\sqrt{100x_i^2 + x_{i+1}^2}\right) - 0.5}{1 + 0.001\left(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2\right)^2}\right)^2$	0
f14	Discus	$[-100, 100]^D$	min $f(x) = 10^6 x_1^2 + \sum_{i=1}^{D} x_i^2$	0
f15	Happy Cat	$[-2, 2]^D$	min $f(x) = \left[(\lvert\lvert x_i^2 \rvert\rvert - D)^2\right]^{\alpha} + \frac{1}{D}(0.5\lvert\lvert x_i^2 \rvert\rvert + \sum_{i=1}^{D} x_i) + 0.5$, where $\alpha = \frac{1}{4}$	0
f16	Drop-Wave Function	$[-5.2, 5.2]^D$	min $f(x) = -\frac{1 + \cos(12\sqrt{x_1^2 + x_2^2})}{(0.5(x_1^2 + x_2^2) + 2)}$	−1
f17	Schaffer 2	$[-100, 100]^D$	min $f(x) = 0.5 + \frac{\sin^2(x_1^2 - x_2^2)^2 - 0.5}{1 + 0.001(x_1^2 + x_2^2)^2}$	0
f18	Camel Function — Three Hump	$[-5, 5]^D$	min $f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1 x_2 + x_2^2$	0

values for both, worst and mean indicators. On average, in simulations with 30-dimensional search space, proposed GOQRFA showed better performance than GDAFA.

Finally, simulations with a search space of 100 dimensions are the most difficult and as expected, proposed GOQRFA and GDAFA proved as two best metaheuristics with the side note that on average GOQRFA showed better performance. In the $f13$ test, GOQRFA for all three indicators — best, worst, and mean, established better results than the GDAFA. Also, in the case of $f1$, $f2$, $f4$, $f7$, $f8$, $f10$, $f12$ and $f15$ simulations, GOQRFA obtained better results for worst and mean metrics. The GOQRFA established better mean value for $f3$ and $f5$ test instance and worst value for $f6$, $f9$ and $f14$. At the other side, GDAFA outscored GOQRFA for worst indicator in $f3$, $f5$ and $f11$ and for the mean indicator in $f6$, $f9$ and 14 instances. Even with relatively large search space, both metaheuristics GDAFA and GOQRFA, in most tests managed to converge to the global optimum and showed good exploratory performance.

In the experiment with two-dimensional functions ($f16$–$f18$), all FA variants reached the global optimum, however when taking into account worst and mean indicators GOQRFA and GDAFA established the best results.

With the goal of better visualizing performance difference between improved FA metaheuristics, summary table for unconstrained simulations is shown, where the number of times each algorithm obtained the best results for each benchmark instance and each performance indicator is counted, (Table 8).

From Table 8, the performance difference between proposed GOQRFA and other improved FA implementations can be clearly noticed. In total, 48 times GOQRFA was better than all other approaches.

Further, to see if there is a statistically significant difference in results, Wilcoxon signed rank-test that performs the pair-wise results' comparison between the proposed GOQRFA and other improved FA's versions and the original FA algorithm for 100-dimensional simulations (Table 6) is applied. Following the usual practice for determining whether the results came from different distributions, significance level of $\alpha = 0.05$ is taken.

The Table 9 summarizes results of applied Wilcoxon test.

The p-value obtained in the test is in all cases < 0.05, which indicates the significant difference between the proposed algorithm and all other compared methods.

Moreover, to better visualize comparative analysis between the GOQRFA and the basic FA, swarm plot diagrams and convergence speed graphs for some test instances are generated. Since the most distinguished performance difference between these two metaheuristics is noticed in the tests with 100 dimensions, swarm plot diagrams for some 100-dimensional functions, where each individual represents the best solution obtained in one run, is shown in Fig. 1.

From the shown swarm plots it can be seen how the original FA in all runs misses the optimum search space in the case of $f4$ and $f12$ benchmark functions. In the case of $f4$, the best solutions generated in all runs are scattered between 436 and 483, which leads to worse mean results. Also, in this test, even in better runs, due to the insufficient exploration power, the FA could not converge to the optimum domain and establishes the best results of around 436, which is far away from the global optimum. On the contrarily, our proposed GOQRFA obtains relatively good results and in all runs managed to converge to the optimum region of the search space. Similarly can be stated for $f12$ benchmark. In all runs, due to the insufficient exploration, the FA completely misses the optimum domain and gets stuck in some of the sub-optimum regions. The GOQRFA on the other hand performs fine exploitation around the global best solutions and manages to find optimum in many runs. Original FA performs better for $f2$ test instance, however, GOAQRFA also in this case obtains much better results.

Additionally, convergence speed graphs of mean results obtained in 50 independent runs for unimodal $f1$ and $f14$ benchmarks and multimodal $f3$ and $f13$ benchmarks with 30 and 100 dimensions are depicted in Figs. 2 and 3, respectively.

In the presented figures can be noticed that at most 300 iterations, GOQRFA converges to the optimum domain of the search space (DUC mechanism) and after it performs a fine-tuned search in the promising regions (IUC mechanism). Also, a clear difference in convergence speed between the original FA and GOQRFA can be noticed. Moreover, from the presented graphs can be observed that in some consecutive iterations basic FA cannot improve solutions and gets stuck in sub-optimal domains of the search space.

Finally, to justify proposed changes and to establish the influence of the GO and QRBL mechanism on the GOQRFA's performance, since both additional procedures introduce overhead in terms of algorithm complexity, additional tests with 100-dimensional benchmarks were conducted. For that purpose, improved FA with only GO and DUC and IUC mechanisms (GOFA) and FA with only QRBL mechanism (QRFA)

Table 4
Comparative analysis with original FA and five other FA's implementations for benchmarks with 10 dimensions.

Function	Algorithm	Best value	Worst value	Mean value	Function	Algorithm	Best value	Worst value	Mean value
	FA	0	2.91e−05	6.07e−07		FA	0	1.00e−03	2.00e−05
	VSSFA	0	0	0		VSSFA	0	0	0
	LFA	0	0.51142	0.142967		LFA	0	0.71557	0.318159
$f_1(x)$	GDAFA	0	0	0	$f_9(x)$	GDAFA	0	0	0
	WFA	8.013e−03	0.11648	0.068951		WFA	6.02e−02	0.79956	0.431151
	CLFA	0	0	0		CLFA	0	0	0
	GOQRFA	0	0	0		GOQRFA	0	0	0
	FA	0	5.08-07	1.06e−08		FA	1.13e−06	3.29e−06	1.17e−06
	VSSFA	0	0	0		VSSFA	0	0	0
	LFA	0	5.4476	1.104484		LFA	0	15.119	2.427798
$f_2(x)$	GDAFA	0	0	0	$f_{10}(x)$	GDAFA	0	0	0
	WFA	1.50e−03	4.0492	1.022306		WFA	4.82e−02	9.2701	3.272067
	CLFA	0	0	0		CLFA	0	0	0
	GOQRFA	0	0	0		GOQRFA	0	0	0
	FA	9.60e−02	9.60e−02	9.60e−02		FA	0	3.02e−07	6.30e−09
	VSSFA	0	0	0		VSSFA	0	0	0
	LFA	0	4.26e−02	6.89e−03		LFA	0	0.40004	0.125577
$f_3(x)$	GDAFA	0	0	0	$f_{11}(x)$	GDAFA	0	0	0
	WFA	7.67e−05	1.75e−02	7.47e−03		WFA	1.96e−02	0.23554	0.110871
	CLFA	0	0	0		CLFA	0	0	0
	GOQRFA	0	0	0		GOQRFA	0	0	0
	FA	5.9697	5.9697	5.9697		FA	−3.0077	−3.0077	−3.00774
	VSSFA	1.1207	16.987	8.676933		VSSFA	−7.4072	−6.4696	−6.68047
	LFA	0	4	2.15633		LFA	−9	−8.2748	−8.82709
$f_4(x)$	GDAFA	0	0	0	$f_{12}(x)$	GDAFA	−9	−9	−9
	WFA	2.3178	21.953	10.60387		WFA	−8.9819	−6.7685	−7.97286
	CLFA	2.1427	27.12	11.42838		CLFA	−7.6738	−5.3186	−6.73002
	GOQRFA	0	0	0		GOQRFA	−9	−9	−9
	FA	0	1.35e−05	2.81e−07		FA	0.5020	0.5020	0.50200
	VSSFA	0	0	0		VSSFA	1.01e−03	1.36e−02	7.24e−03
	LFA	0	1.1064	0.361658		LFA	0	8.79e−03	1.81e−03
$f_5(x)$	GDAFA	0	0	0	$f_{13}(x)$	GDAFA	0	0	0
	WFA	1.15e−02	0.64831	0.303216		WFA	9.71e−04	2.29e−02	1.01e−02
	CLFA	0	0	0		CLFA	1.54e−04	2.41e−02	8.64e−03
	GOQRFA	0	0	0		HFA	0	0	0
	FA	1.46e−14	1.90e−03	3.8e−05		FA	643.0253	697.9746	644.1241
	VSSFA	8.88e−16	8.88e−16	8.88e−16		VSSFA	0	0	0
	LFA	8.88e−16	1.1568	0.378197		LFA	0	0.53477	6.27e−02
$f_6(x)$	GDAFA	8.88e−16	8.88e−16	8.88e−16	$f_{14}(x)$	GDAFA	0	0	0
	WFA	7.75e−02	1.0976	0.578406		WFA	2.50e−03	0.16738	7.23e−02
	CLFA	8.88e−16	8.88e−16	8.88e−16		CLFA	0	0	0
	GOQRFA	8.88e−16	8.88e−16	8.88e−16		GOQRFA	0	0	0
	FA	1.61e−07	1.83e−07	1.62e−07		FA	1.7538	1.7538	1.75380
	VSSFA	0	0	0		VSSFA	0	0.4	4.47e−02
	LFA	0	1.02e−02	1.44e−03		LFA	0	0.4	0.146667
$f_7(x)$	GDAFA	0	0	0	$f_{15}(x)$	GDAFA	0	0	0
	WFA	4.938e−13	3.45e−03	3.51e−04		WFA	0.61011	0.98881	0.85317
	CLFA	0	0	0		CLFA	0	0.6	0.16
	GOQRFA	0	0	0		GOQRFA	0	0	0
	FA	0	1.08e−06	2.24e−08					
	VSSFA	0	0	0					
	LFA	0	1.4135	0.217733					
$f_8(x)$	GDAFA	0	0	0					
	WFA	1.69e−02	0.7670	0.346934					
	CLFA	0	0	0					
	GOQRFA	0	0	0					

were implemented. Both versions utilize dynamic parameter α. Testing results for 100-dimensional problems are shown in Table 10.

Results from the presented table clearly indicate that GOQRFA establishes better performance than GOFA and QRFA. Also, when compared with original FA (Table 6), GOFA and QRFA obtained better solutions' quality for 100-dimensional instances.

4.2. Feature selection simulations

In this subsection, metaheuristic adaptations for feature selection challenge along with fitness function formulation are shown first, followed by evaluation metrics, datasets description, control parameter setup and comparative analysis.

4.2.1. Solutions' encoding strategy, initialization scheme and fitness formulation

For tackling feature selection problem with proposed GOQRFA, the method should be adapted for binary combinatorial optimization challenges. Each GOQRFA solution represents a set of features in the dataset of a length N, where N denotes the total number of attributes in a dataset, with the search space of dimension 2^N. If the value of solution at position j is 1, the feature at this position will be included and if it is 0, then this feature will be excluded from classification. Adapted approach is refereed as bGOQRFA.

To encode a candidate solutions as a N-bit string, different transfer functions from S-shape and V-shape families can be used [55]. However, after extensive empirical experiments with different transfer

Table 5
Comparative analysis with original FA and five other FA's implementations for benchmarks with 30 dimensions.

Function	Algorithm	Best value	Worst value	Mean value	Function	Algorithm	Best value	Worst value	Mean value
	FA	0	3.30e−04	8.47e−06		FA	0	7.15e−03	7.43e−04
	VSSFA	13.616	20.068	17.05007		VSSFA	15.568	18.532	17.28833
	LFA	9.6426	17.403	14.08179		LFA	7.996	13.552	11.55308
$f_1(x)$	GDAFA	0	1.81e−04	1.61e−05	$f_9(x)$	GDAFA	0	**4.84e−03**	**6.22e−04**
	WFA	0.14182	0.31932	0.237264		WFA	0.75795	1.877	1.444582
	CLFA	8.89e−02	0.34271	0.194190		CLFA	0.67279	2.0693	1.444582
	GOQRFA	0	**7.15E−05**	**2.31E−06**		GOQRFA	0	6.17e−03	7.25e−04
	FA	0	5.73e−04	1.19e−05		FA	5.52e−04	1.48e−03	5.71e−04
	VSSFA	949.73	1332.7	1163.641		VSSFA	740.56	4302.5	3334.035
	LFA	495.23	1030.9	863.976		LFA	1397.7	3485.4	2736.29
$f_2(x)$	GDAFA	0	6.01e−04	3.53e−05	$f_{10}(x)$	GDAFA	0	1.49e−02	1.28e−03
	WFA	9.0823	27.288	17.18261		WFA	3.5447	31.875	23.61039
	CLFA	8.4382	34.836	18.24006		CLFA	5.3022	29.98	19.11064
	GOQRFA	0	**4.56e−05**	**3.32e−06**		GOQRFA	0	**1.13e−03**	**4.44e−04**
	FA	9.86e−03	9.87e−03	9.86e−03		FA	1.66e−02	1.86e−02	1.66e−02
	VSSFA	0.47304	0.583	0.537958		VSSFA	13.113	16.154	14.56793
	LFA	0.34297	0.51144	0.43649		LFA	6.508	12.429	10.71938
$f_3(x)$	GDAFA	0	**1.84e−05**	3.65e−06	$f_{11}(x)$	GDAFA	0	3.34e−03	4.69e−04
	WFA	5.47e03	1.90e−02	1.23e−02		WFA	0.24505	0.61146	0.408792
	CLFA	4.85e−03	1.76e−02	9.86e−03		CLFA	0.11367	0.46509	0.301099
	GOQRFA	0	5.69e−05	**2.11e−06**		GOQRFA	0	**9.94e−04**	**1.05e−04**
	FA	53.728	53.728	53.728		FA	−2.74	−2.738	−2.7380
	VSSFA	90.168	147.03	133.9519		VSSFA	−14.271	−10.216	−11.661
	LFA	26.942	46.808	38.35837		LFA	−19.773	−15.487	−17.0816
$f_4(x)$	GDAFA	0	0.29587	6.19e−02	$f_{12}(x)$	GDAFA	−29	**−28.98**	−28.9957
	WFA	10.562	70.987	42.79867		WFA	−27.14	−23.372	−25.35393
	CLFA	46.82	116.4	91.65793		CLFA	−19.932	−13.57	−16.3889
	GOQRFA	0	**0.15332**	**2.11e−02**		GOQRFA	−29	−28.97	−28.9957
	FA	0	8.10e−04	1.69e−05		FA	4.82	4.854	4.8218
	VSSFA	189.9	259.87	228.3503		VSSFA	4.96e−02	9.17e−02	7.34e−02
	LFA	173.79	249.66	217.1767		LFA	4.92e−02	0.1031	7.61e−02
$f_5(x)$	GDAFA	0	1.30e−04	1.31e−05	$f_{13}(x)$	GDAFA	2.77e−32	8.85e−06	1.08e−06
	WFA	1.0638	5.8679	3.364743		WFA	4.38e−02	0.11132	7.60e−02
	CLFA	1.177	7.1741	3.68803		CLFA	3.61e−02	9.92e−02	7.06e−02
	GOQRFA	0	**7.15e−05**	**6.65e−06**		GOQRFA	**4.34e−34**	**6.13e−06**	**7.09e−07**
	FA	3.95e−14	4.08e−02	8.17e−03		FA	1.26e+04	1.29e+04	1.26e+04
	VSSFA	4.1408	4.4306	4.29383		VSSFA	16.331	22.498	19.8985
	LFA	3.0118	3.7251	3.39331		LFA	12.645	20.226	16.6886
$f_6(x)$	GDAFA	8.88e−16	1.91e−02	**2.97e−03**	$f_{14}(x)$	GDAFA	0	3.01e−04	4.50e−05
	WFA	0.42363	0.86514	0.698177		WFA	3.54e−02	0.50956	0.261277
	CLFA	0.44769	2.8507	1.1317517		CLFA	0.10623	0.42646	0.244281
	GOQRFA	8.88e−16	**1.15e−02**	4.65e−03		GOQRFA	0	**1.15e−04**	**2.29e−05**
	FA	6.16e−05	6.18e−05	6.16e−05		FA	2.3302	2.3302	2.3302
	VSSFA	123.29	1158.4	341.4773		VSSFA	2.2622	2.348	2.30503
	LFA	21.96	4339.5	1065.334		LFA	0.73333	1.4667	1.060788
$f_7(x)$	GDAFA	0	1.17e−38	3.91e−40	$f_{15}(x)$	GDAFA	0	0.62705	**9.97e−02**
	WFA	4.47E−11	1.29e−03	9.667E−05		WFA	1.1758	1.3716	1.29469
	CLFA	3.526e−08	9.96e−04	1.03e−03		CLFA	1.3084	1.744	1.534977
	GOQRFA	0	**5.91e−39**	**3.02e−41**		GOQRFA	0	**0.59256**	9.98e−02
	FA	0	3.17e−04	8.12e−05					
	VSSFA	153.83	274.97	233.8543					
	LFA	100.28	214.06	175.337					
$f_8(x)$	GDAFA	0	5.81e−04	4.72e−05					
	WFA	1.524	4.7709	3.25577					
	CLFA	1.561	5.1752	3.2706972					
	GOQRFA	0	**2.22e−04**	**9.29e−06**					

functions, it is determined that by using standard Sigmoidal transfer function, on average the best performance is obtained. By applying this function for each candidate solution i and parameter j continuous values from the range $[0, 1]$ are mapped to either 0 or 1:

$$x_{i,j} = \begin{cases} 1, & \text{if} \quad 1/(1 + e^{-x_{i,j}}) > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

The initialization phase is very important in any metaheuristics because from the chosen initialization strategy the quality of final solutions depend. Recently, Xue et al. [7] have proposed small, large and mixed initialization strategy for feature selection. Small and large strategies generate initial population where for each individual most of the features are disabled (zero) and enabled (one), respectively. Mixed strategy combines properties of small and large initialization where most individuals are initialized by using small, while remaining by employing large strategy. Again, by performing practical simulations with mentioned three initialization strategies plus traditional strategy (where each solution is completely randomly generated), it is determined that the mixed strategy on average generates better solutions and these results are included in comparative analysis.

In the proposed research, for determining efficiency of a method, classification accuracy along with the number of selected features were taken into the account. Thus, in this case two contradictory objectives should be minimized and the following composite objective function is

Table 6
Comparative analysis with original FA and five other FA's implementations for benchmarks with 100 dimensions.

Function	Algorithm	Best value	Worst value	Mean value	Function	Algorithm	Best value	Worst value	Mean value
	FA	0	5.08e−03	3.06e−04		FA	7.27	7.345	7.2724
	VSSFA	86.407	95.975	91.5817		VSSFA	74.271	79.171	76.217
	LFA	83.653	100.5	94.2248		LFA	61.281	69.287	65.83797
$f_1(x)$	GDAFA	0	5.23e−03	1.66e−03	$f_9(x)$	GDAFA	0	0.27298	**6.85e−02**
	WFA	0.77697	1.3239	1.021821		WFA	4.668	7.2477	5.95169
	CLFA	6.0074	9.0963	7.497247		CLFA	19.463	29.886	24.39307
	GOQRFA	0	**3.39e−03**	**2.13e−04**		GOQRFA	0	**0.253**	8.13e−02
	FA	6.45e−02	0.8613	0.2963		FA	0.11	0.764	0.2127
	VSSFA	20155	23097	21435.67		VSSFA	24876	29942	27295.17
	LFA	20134	23511	22061.73		LFA	24626	33338	29210.1
$f_2(x)$	GDAFA	0	2.7	0.263	$f_{10}(x)$	GDAFA	0	0.76923	6.21e−02
	WFA	183.59	326.04	247.351		WFA	92.637	143.93	117.6095
	CLFA	1435.2	2495	1958.367		CLFA	686.78	1421.7	1057.672
	GOQRFA	0	**0.525**	**0.255**		GOQRFA	0	**0.731**	**5.44e−02**
	FA	1.11e−16	2.99e−04	5.97e−05		FA	14.77	15.145	14.7801
	VSSFA	0.78385	0.85044	0.81136		VSSFA	67.41	73.173	70.5776
	LFA	0.74602	0.83769	0.799866		LFA	58.349	65.707	62.45623
$f_3(x)$	GDAFA	0	**1.24e−04**	3.50e−05	$f_{11}(x)$	GDAFA	0	**2.25e−02**	**4.53e−03**
	WFA	1.17e−02	2.30e−02	2.10e−02		WFA	1.2958	2.2723	1.67696
	CLFA	0.10731	0.19698	0.166489		CLFA	7.2875	11.657	9.200357
	GOQRFA	0	1.38e−04	**2.14e−05**		GOQRFA	0	2.41e−02	6.02e−03
	FA	436.88	551.39	484.60		FA	−4.44	−8.728	−6.1785
	VSSFA	638.51	706.69	668.54		VSSFA	−23.042	−19.167	−20.9113
	LFA	220.18	260.46	241.692		LFA	−42.456	−33.023	−36.48
$f_4(x)$	GDAFA	0	1.6535	0.567833	$f_{12}(x)$	GDAFA	−99	−98.845	−98.9479
	WFA	112.5	214.36	179.406		WFA	−87.93	−79.334	−82.79143
	CLFA	478.93	603.43	556.4503		CLFA	−40.354	−29.746	−36.3625
	GOQRFA	0	**1.193**	**0.541**		GOQRFA	−99	**−98.882**	**−98.963**
	FA	2428.9	2592.2	2432.19		FA	20.65	20.770	20.6599
	VSSFA	4002.6	4633	4310.623		VSSFA	0.14201	0.16768	0.154435
	LFA	3878.5	4681.6	4362.873		LFA	0.1476	0.19154	0.175026
$f_5(x)$	GDAFA	0	**0.41709**	8.17e−02	$f_{13}(x)$	GDAFA	1.23e−32	8.70e−05	1.30e−05
	WFA	30.682	69.406	50.59657		WFA	0.13246	0.17611	0.155534
	CLFA	289	508.96	384.6423		CLFA	0.1269	0.16589	0.152831
	GOQRFA	0	0.425	**4.44e−02**		GOQRFA	**3.15e−33**	**3.33e−05**	**9.45e−06**
	FA	1.11e−13	9.43e−02	1.89e−02		FA	1.70e+0.5	1.72e+05	1.70e+05
	VSSFA	4.9241	5.1426	5.05314		VSSFA	91.691	104.63	99.2656
	LFA	4.2498	4.5533	4.413013		LFA	94.901	108.04	102.4729
$f_6(x)$	GDAFA	8.88e−16	3.39e−02	**1.08e−02**	$f_{14}(x)$	GDAFA	0	3.06e−03	**6.23e−04**
	WFA	0.6256	1.0087	0.837799		WFA	0.89019	1.4583	1.099807
	CLFA	3.1695	4.2067	3.612709		CLFA	8.7565	15.82	11.24761
	GOQRFA	8.88e−16	**2.95e−02**	1.23e−02		GOQRFA	0	**2.75e−03**	6.41e−04
	FA	7.30e−05	7.31e−05	7.35e−05		FA	3.1582	3.1583	3.158275
	VSSFA	1.01e+15	6.59e+17	1.29e+17		VSSFA	3.1769	3.2378	3.221253
	LFA	1.70e+18	1.32e+22	1.88e+21		LFA	2.2808	2.4819	2.405193
$f_7(x)$	GDAFA	0	2.34e−31	8.11e−33	$f_{15}(x)$	GDAFA	0	0.79772	0.441660
	WFA	3.017e−09	2.16e−03	1.17e−04		WFA	1.7363	1.8669	1.812157
	CLFA	219780	2.2613e+11	9.54e+09		CLFA	2.5784	2.8293	2.71441
	GOQRFA	0	**1.55e−32**	**7.81e−34**		GOQRFA	0	**0.735**	**0.428**
	FA	0.18	0.499	0.1890					
	VSSFA	4100.5	4.53e+03	4.33e+03					
	LFA	3541.9	4.79e+03	4.33e+03					
$f_8(x)$	GDAFA	0	0.53765	5.42e−02					
	WFA	36.905	72.045	51.45037					
	CLFA	291.85	487.54	370.4313					
	GOQRFA	0	**0.495**	**3.09e−02**					

utilized to evaluate each solution i from the population:

$$\min O_i = \omega \cdot E_i + (1-\omega) \cdot \frac{\sum_{j=0}^{N} x_{i,j}}{N}, \tag{16}$$

where E_i is the classification error rate, $\sum_{j=0}^{N} x_{i,j}$ denotes number of selected features for solution i, and N is the total number of attributes in the dataset. Continuous parameter $\omega \in [0,1]$ controls the relative influence of the error rate and the number of selected features to the fitness function.

It is noted that since in experiments the goal is to optimize objective function, explicitly defined fitness is not formulated, so the fitness and objective function are used interchangeably.

Fitness function is calculated by employing k-nearest neighbors (KNN) classifier. When solution is generated, it is first being transform to binary array and then features from the specific dataset, that correspond to solution, are extracted and passed for KNN classification. Thus, for each fitness evaluation, the KNN is trained and validated.

4.2.2. Evaluation metrics, datasets description and control parameters' setup

Proposed method is validated against 21 well-known UCL datasets [56], that have small, medium and large number of attributes (features), which are described in Table 11. For each experiment, observations are divided randomly into training, testing and validation subsets

Table 7
Comparative analysis with original FA and five other FA's implementations for benchmarks with 2 dimensions.

Function	Algorithm	Best value	Worst value	Mean value	Function	Algorithm	Best value	Worst value	Mean value
	FA	−1	−1	−1		FA	0	9.72e−13	1.94e−14
	VSSFA	−1	−1	−1		VSSFA	0	0	0
	LFA	−1	−1	−1		LFA	0	0	0
$f_{16}(x)$	GDAFA	−1	−1	−1	$f_{17}(x)$	GDAFA	0	0	0
	WFA	−1	−0.94357	−0.996534		WFA	0	0	0
	CLFA	−1	−1	−1		CLFA	0	0	0
	GOQRFA	−1	−1	−1		GOQRFA	0	0	0
	FA	0	3.02e−12	6.29e−14					
	VSSFA	0	0	0					
	LFA	0	0	0					
$f_{18}(x)$	GDAFA	0	0	0					
	WFA	0	0.00000505	1.861e−07					
	CLFA	0	0	0					
	GOQRFA	0	0	0					

Table 8
Summary of results for unconstrained benchmarks.

	FA	VSSFA	LFA	WFA	CLFA	GDAFA	GOQRFA
Best 10dim	0	0	0	0	0	0	0
Mean 10dim	0	0	0	0	0	0	0
Worst 10dim	0	0	0	0	0	0	0
Total 10 dim	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Best 30dim	0	0	0	0	0	0	1
Mean 30dim	0	0	0	0	0	3	11
Worst 30dim	0	0	0	0	0	3	12
Total 30 dim	**0**	**0**	**0**	**0**	**0**	**6**	**24**
Best 100dim	0	0	0	0	0	0	1
Mean 100dim	0	0	0	0	0	4	11
Worst 100dim	0	0	0	0	0	3	12
Total 100 dim	**0**	**0**	**0**	**0**	**0**	**7**	**24**
GRAND TOTAL	**0**	**0**	**0**	**0**	**0**	**13**	**48**

Table 9
Statistical comparison of the GOQRFA for 100-dimensional tests with other approaches by Wilcoxon Signed-Rank Test ($\alpha = 0.05$).

Function	GOQRFA	GDAFA	FA	VSSFA	LFA	WFA	CLFA
f1	2.13E−04	1.66E−03	3.06E−04	9.16E+01	9.42E+01	1.02E+00	7.50E+00
f2	2.55E−01	2.63E−01	2.96E−01	2.14E+04	2.21E+04	2.47E+02	1.96E+03
f3	2.14E−05	3.50E−05	5.97E−05	8.11E−01	8.00E−01	2.10E−02	1.66E−01
f4	5.41E−01	5.68E−01	4.85E+02	6.69E+02	2.42E+02	1.79E+02	5.56E+02
f5	4.44E−02	8.17E−02	2.43E+03	4.31E+03	4.36E+03	5.06E+01	3.85E+02
f6	1.23E−02	1.08E−02	1.89E−02	5.05E+00	4.41E+00	8.38E−01	3.61E+00
f7	7.81E−34	8.11E−33	7.35E−05	1.29E+17	1.88E+21	1.17E−04	9.54E+09
f8	3.09E−02	5.42E−02	1.89E−01	4.33E+03	4.33E+03	5.15E+01	3.70E+02
f9	8.13E−02	6.85E−02	7.27E+00	7.62E+01	6.58E+01	5.95E+00	2.44E+01
f10	5.44E−02	6.21E−02	2.13E−01	2.73E+04	2.92E+04	1.18E+02	1.06E+03
f11	6.02E−03	4.53E−03	1.48E+01	7.06E+01	6.25E+01	1.68E+00	9.20E+00
f12	−9.90E+01	−9.89E+01	−6.18E+00	−2.09E+01	−3.65E+01	−8.28E+01	−3.64E+01
f13	9.45E−06	1.30E−05	2.07E+01	1.54E−01	1.75E−01	1.56E−01	1.53E−01
f14	6.41E−04	6.23E−04	1.70E+05	9.93E+01	1.02E+02	1.10E+00	1.12E+01
f15	4.28E−01	4.42E−01	3.16E+00	3.22E+00	2.41E+00	1.81E+00	2.71E+00
p-value		**3.19E−02**	**3.05E−05**	**3.05E−05**	**3.05E−05**	**3.05E−05**	**3.05E−05**

by using $train_test_split$ method. The 80% of observations are used for training, while remaining 20% are utilized for testing. Moreover, 10% of training data is used for validation.

In the proposed research, to make comparative analysis with other approaches more realistic, the same metric are used and the same experimental environment as in [57] was set. For validation and comparative analysis the following metrics were used: classification accuracy, objective function value and number of selected features averaged over 20 independent runs.

The average classification accuracy determines average accuracy obtained for each datasets for 20 runs:

$$Avg_{acc} = \frac{1}{Run}\sum_{i=1}^{Run}\frac{1}{N}\sum_{i=1}^{K} Match(\hat{Y}_i, Y_i), \tag{17}$$

where Run denotes the total number of independent runs, K is the number of observations in the test set, while $\hat{Y}_i$ and Y_i denote classifier output and actual class for instance i.

The average objective function value (Avg_o) is calculated as:

$$Avg_o = \frac{1}{Run}\sum_{i=1}^{Run} O_i, \tag{18}$$

where O_i represents the best objective function value obtained in each run.

Finally, the average number of selected features (selection size) is determined as:

$$Avg_{nf} = \frac{1}{Run}\sum_{i=1}^{Run} FS_i, \tag{19}$$

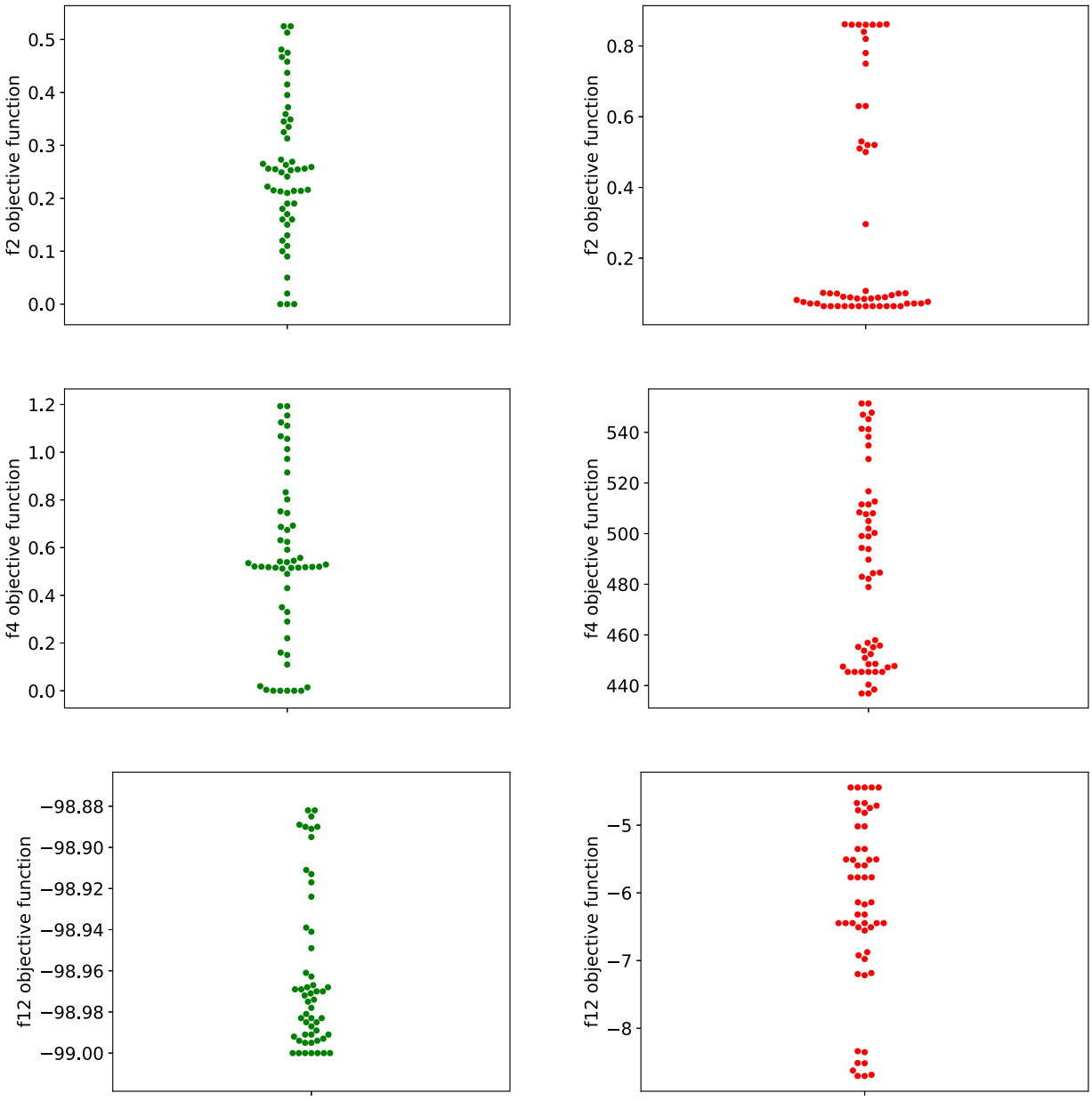


Fig. 1. Swarm plot diagrams — GOQRFA (green) vs. FA (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where FS_i represents the number of selected features of best solution for run i.

All approaches in [57] were tested with 8 individuals in population and 70 iterations (approximately 560 fitness function evaluations — FFE). However, the QRL mechanism of GOQRFA uses additional FFE in each iteration for each solution, and to make comparison realistic, the GOQRFA was tested with only 4 individuals. Other GOQRFA control parameters were set as shown in Table 1.

Results reported in comparative analysis are generated by using $\omega = 0.99$, as it is reported in the manuscript from which results of other methods included in comparative analysis were obtained [57]. Moreover, it is determined that the $k = 5$ is optimum value for KNN classifier and this value is used in simulations.

4.2.3. Comparative analysis and discussion

Comparative analysis was performed with 9 other state-of-the-art metaheuristics for feature selection: WOA (whale optimization algorithm), bWOA-S, bWOA-v, BALO1 (ant lion optimizer), BALO2, BALO3, PSO, bGWO, and bDA, which results are retrieved from [57]. In comparative analysis were also included original binary FA and previously proposed FA with quasi-reflective learning initialization (bFAQRL) [14].

As noted above, all results are generated by using mixed initialization strategy and average objective function value (fitness), accuracy and number of selected features for 20 independent runs are used as metrics for performance comparisons. Comparative analysis for these three indicators is reported in Tables 12, 13 and 14, respectively. The convergence graph, for selected datasets, of the proposed method in feature selection experiment are depicted in Fig. 4.

It is observed that results reported for average number of selected features in [57] are wrong — the authors accidentally reported results which are 10 times smaller than real. Even with corrections, results reported in [57] are strange because optimizers employed relatively small number of features. For this set of experiments authors have probably used fitness function, where higher influence has the number of features

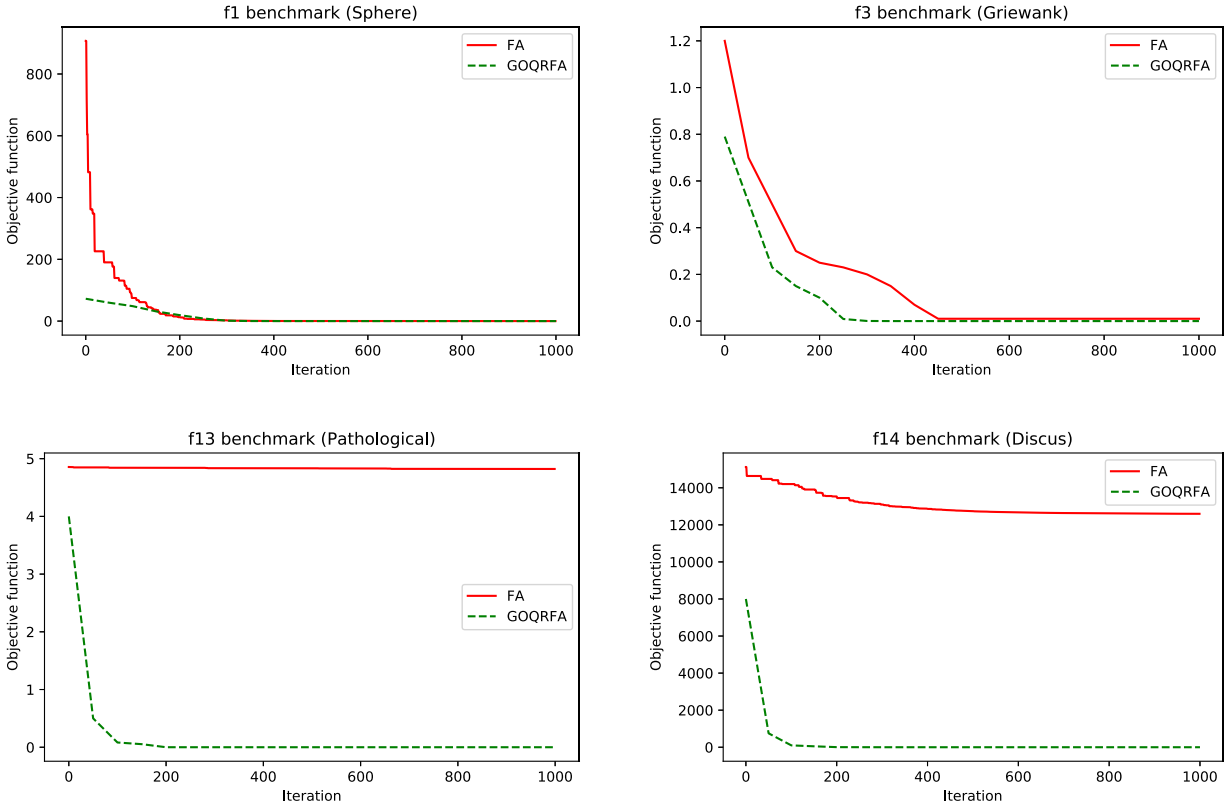


Fig. 2. Convergence speed graphs for 30-dimensional benchmarks — GOQRFA vs. FA.

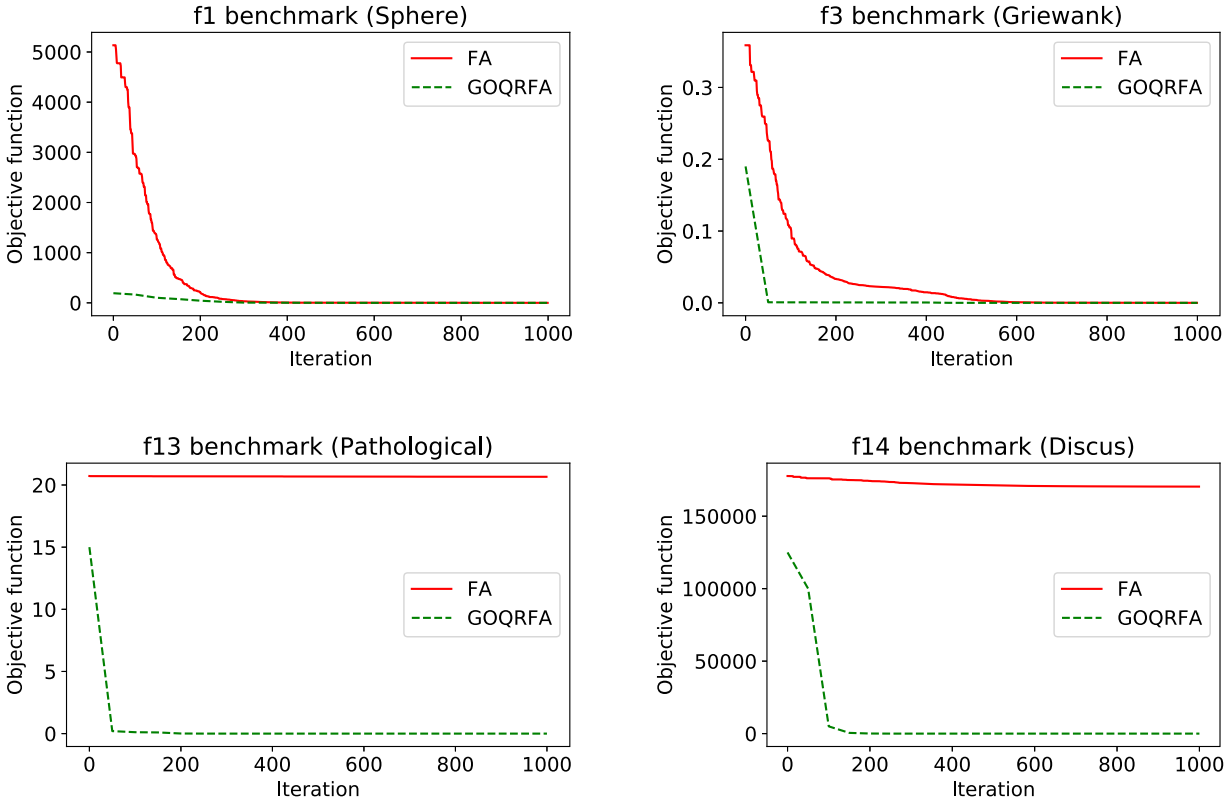


Fig. 3. Convergence speed graphs for 100-dimensional benchmarks — GOQRFA vs. FA.

Table 10
Comparative analysis between proposed approach and GOFA and QRFA with 100-dimensional benchmarks.

Function	Algorithm	Best value	Worst value	Mean value	Function	Algorithm	Best value	Worst value	Mean value
$f_1(x)$	GOFA	0	4.98e−03	3.03e−04	$f_9(x)$	GOFA	6.515	7.945	7.201
	QRFA	0	4.46e−03	2.82e−04		QRFA	5.925	7.821	6.372
	GOQRFA	0	**3.39e−03**	**2.13e−04**		GOQRFA	**0**	**0.253**	**8.13e−02**
$f_2(x)$	GOFA	0.32	0.725	0.238	$f_{10}(x)$	GOFA	0.06	0.755	0.203
	QRFA	0.19	0.692	0.205		QRFA	0.03	0.742	0.189
	GOQRFA	**0**	**0.525**	**0.255**		GOQRFA	**0**	**0.731**	**5.44e−02**
$f_3(x)$	GOFA	0	2.85e−04	4.85e−05	$f_{11}(x)$	GOFA	6.55	10.11	8.54
	QRFA	0	2.44e−04	3.68e−05		QRFA	4.29	8.08	6.71
	GOQRFA	0	**1.38e−04**	**2.14e−05**		GOQRFA	**0**	**2.41e−02**	**6.02e−03**
$f_4(x)$	GOFA	432.78	551.56	419.71	$f_{12}(x)$	GOFA	−40.701	−31.555	−37.661
	QRFA	295.37	365.82	321.47		QRFA	−61.375	−52.752	−57.015
	GOQRFA	**0**	**1.193**	**0.541**		GOQRFA	**−99**	**−98.882**	**−98.963**
$f_5(x)$	GOFA	1275.21	1409.73	1344.02	$f_{13}(x)$	GOFA	0.1152	0.1490	0.1309
	QRFA	973.97	1265.43	1119.54		QRFA	0.1003	0.1351	0.1186
	GOQRFA	**0**	**0.425**	**4.44e−02**		GOQRFA	**3.15e−33**	**3.33e−05**	**9.45e−06**
$f_6(x)$	GOFA	3.23e−14	8.79e−02	1.56e−02	$f_{14}(x)$	GOFA	8.3325	14.2301	10.5972
	QRFA	8.92e−15	6.66e−02	1.41e−2		QRFA	6.7251	8.1796	7.5403
	GOQRFA	**8.88e−16**	**2.95e−02**	**1.23e−02**		GOQRFA	**0**	**2.75e−03**	**6.41e−04**
$f_7(x)$	GOFA	4.23e−06	3.89e−04	7.13e−05	$f_{15}(x)$	GOFA	2.6934	2.9134	2.7921
	QRFA	8.75e−07	2.29e−05	4.42e−06		QRFA	1.9582	1.9845	1.9763
	GOQRFA	**0**	**1.55e−32**	**7.81e−34**		GOQRFA	**0**	**0.735**	**0.428**
$f_8(x)$	GOFA	0.17	0.497	0.183					
	QRFA	0.09	0.497	0.162					
	GOQRFA	**0**	**0.495**	**3.09e−02**					

Table 11
List of datasets used in the experiments results.

No.	Name	Features	Samples
1	Breastcancer	9	699
2	Tic-tac-toe	9	958
3	Zoo	16	101
4	WineEW	13	178
5	SpectEW	22	267
6	SonarEW	60	208
7	IonosphereEW	34	351
8	HeartEW	13	270
9	CongressEW	16	435
10	KrvskpEW	36	3196
11	WaveformEW	40	5000
12	Exactly	13	1000
13	Exactly 2	13	1000
14	M-of-N	13	1000
15	vote	16	300
16	BreastEW	30	569
17	Semeion	265	1593
18	Clean 1	166	476
19	Clean 2	166	6598
20	Lymphography	18	148
21	PenghungEW	325	73

than classification accuracy, which is not reported in [57]. however in table provided in this manuscript, these results are corrected.

Additionally, the Friedman ranks and Holm's procedure is applied, and the results proves the robustness of the proposed method (see Tables 15 and 16).

From presented comparative analysis interesting conclusions can be deduced. However, most importantly, proposed bGOQRFA in average shows the best performance than all other adversary approaches. Moreover, as mentioned above, it is noticed that all three FA approaches implemented for the purpose of this research obtain significantly higher number of features for most datasets than algorithms which results are reported in [57]. From composite objective (fitness) function formulation Eq. (16), it is obvious that with $\omega = 0.99$, the higher impact on the objective has classification error (accuracy) than the number of features and that is the main argument why reported results for implemented FA approaches have relatively high number of features.

Also, when compared with basic FA and recently proposed bFAQRLI 1u [14], the bGOARFA in most benchmarks obtains lowest fitness value and highest classification accuracy. However, there is always a trade-off. For example, when performing feature selection with *SonarEW*, *Clean2* and *ExactlyEW* datasets, propose bGOQRFA establishes worse performance for accuracy, as well as for the fitness than basic bFA and bFAQRLI. Similarly, when comparing results with *WaveFormEW* and *Vote* test instances, bFAQRLI proved as more robust method than the one proposed in this research. There is also a case when basic FA outperforms both improved FA implementation (*KrivskpEW* test instance).

In general, all three FA instances in average show better results than all other methods encompassed in comparative analysis. The only exclusion is *Semion* dataset, where for all three indicators other methods outscored FA-based metaheuristics. The comparison of computational time is not provided because other metaheuristics were tested on a different platform. However, by reducing number of features, the classification process is significantly reduced regardless of the platform which is used for execution.

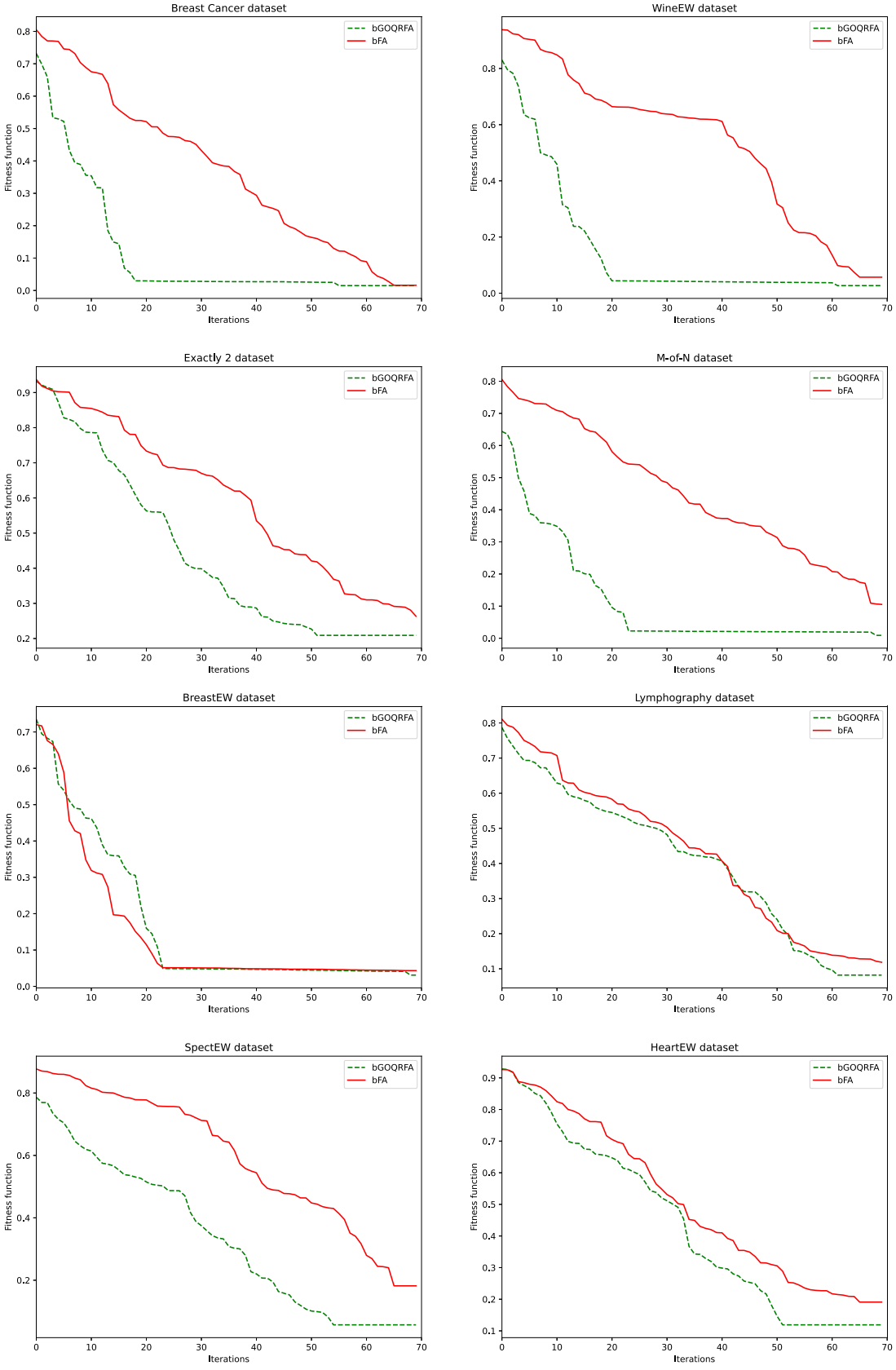


Fig. 4. Convergence graphs for benchmark datasets.

Table 12
Fitness (objective) value for 21 datasets averaged over 20 runs.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA	bFA	bFAQRLI	bGOQRFA
1	0.054	0.052	0.079	0.100	0.099	0.076	0.031	0.035	0.032	0.016	**0.013**	0.015
2	0.220	0.207	0.215	0.245	0.252	0.246	0.204	0.215	0.209	0.219	0.180	**0.175**
3	0.153	0.148	0.120	0.183	0.146	0.141	0.078	0.096	0.071	0.054	**0.013**	0.021
4	0.925	0.928	0.910	0.935	0.938	0.938	0.884	0.903	0.882	0.057	0.029	**0.027**
5	0.313	0.307	0.289	0.319	0.321	0.312	0.242	0.280	0.255	0.182	0.064	**0.057**
6	0.304	0.286	0.254	0.278	0.298	0.285	0.168	0.235	0.194	0.073	**0.059**	0.085
7	0.159	0.158	0.152	0.156	0.169	0.165	0.113	0.141	0.124	0.084	0.106	**0.071**
8	0.328	0.308	0.259	0.319	0.324	0.308	0.158	0.233	0.167	0.191	0.125	**0.119**
9	0.389	0.380	0.372	0.393	0.397	0.384	0.337	0.359	0.341	0.053	0.028	**0.024**
10	0.071	0.074	0.081	0.074	0.072	0.074	0.040	0.061	0.053	**0.028**	0.032	0.032
11	0.193	0.193	0.195	0.198	0.195	0.193	0.182	0.187	0.188	0.171	**0.157**	0.162
12	0.303	0.308	0.301	0.301	0.307	0.308	0.151	0.272	0.226	0.228	**0.099**	0.233
13	0.241	0.244	0.252	0.237	0.244	0.253	0.238	0.244	0.243	0.252	0.225	**0.209**
14	0.139	0.133	0.155	0.151	0.150	0.136	0.022	0.112	0.072	0.095	0.010	**0.009**
15	0.084	0.084	0.081	0.089	0.090	0.085	0.048	0.069	0.052	0.060	**0.038**	0.048
16	0.081	0.058	0.062	0.086	0.088	0.086	0.033	0.057	0.031	0.033	0.056	**0.031**
17	0.044	0.043	0.037	0.043	0.043	0.044	0.032	0.034	**0.030**	0.084	0.066	0.092
18	0.191	0.187	0.176	0.184	0.192	0.197	0.136	0.158	0.149	0.089	0.075	**0.071**
19	0.052	0.052	0.049	0.051	0.052	0.052	0.041	0.044	0.042	0.036	**0.025**	0.042
20	0.235	0.230	0.223	0.258	0.243	0.237	0.138	0.211	0.160	0.108	0.111	**0.082**
21	0.260	0.244	0.242	0.276	0.262	0.274	0.149	0.217	0.180	0.136	0.071	**0.062**
Total	0	0	0	0	0	0	0	0	1	1	7	12

Table 13
Classification accuracy for 21 datasets averaged over 20 runs.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA	bFA	bFAQRLI	bGOQRFA
1	0.785	0.619	0.628	0.740	0.725	0.726	0.802	0.962	0.789	0.988	0.992	**0.995**
2	0.787	0.799	0.786	0.686	0.681	0.686	0.720	0.764	0.673	0.784	0.824	**0.829**
3	0.841	0.839	0.822	0.656	0.706	0.680	0.789	0.900	0.779	0.949	**0.989**	0.982
4	0.065	0.056	0.053	0.039	0.033	0.031	0.039	0.086	0.031	0.946	0.973	**0.975**
5	0.678	0.670	0.664	0.635	0.623	0.625	0.656	0.707	0.649	0.819	0.938	**0.945**
6	0.698	0.703	0.703	0.645	0.639	0.647	0.721	0.765	0.705	0.928	**0.942**	0.916
7	0.835	0.836	0.831	0.819	0.803	0.802	0.835	0.860	0.827	0.918	0.894	**0.931**
8	0.656	0.654	0.652	0.625	0.621	0.623	0.668	0.751	0.652	0.811	0.879	**0.885**
9	0.598	0.582	0.595	0.573	0.559	0.577	0.589	0.631	0.571	0.950	0.974	**0.981**
10	0.936	0.930	0.918	0.766	0.765	0.757	0.794	0.943	0.754	**0.976**	0.971	0.972
11	0.812	0.808	0.804	0.642	0.649	0.647	0.763	0.816	0.747	0.832	**0.847**	0.839
12	0.687	0.683	0.691	0.644	0.656	0.648	0.664	0.706	0.642	0.775	**0.904**	0.772
13	0.738	0.740	0.735	0.733	0.711	0.703	0.723	0.735	0.712	0.750	0.778	**0.796**
14	0.865	0.865	0.833	0.734	0.732	0.744	0.761	0.883	0.728	0.908	0.992	**0.994**
15	0.915	0.908	0.900	0.829	0.823	0.829	0.884	0.930	0.866	0.943	**0.968**	0.955
16	0.761	0.610	0.615	0.730	0.744	0.727	0.810	0.944	0.769	0.972	0.949	**0.976**
17	0.964	0.964	0.965	0.924	0.939	0.925	0.956	**0.972**	0.959	0.918	0.934	0.910
18	0.815	0.818	0.803	0.729	0.720	0.724	0.806	0.845	0.791	0.913	0.926	**0.931**
19	0.956	0.956	0.955	0.908	0.910	0.911	0.953	0.962	0.952	0.967	**0.979**	0.965
20	0.756	0.755	0.749	0.639	0.672	0.659	0.705	0.786	0.709	0.899	0.889	**0.925**
21	0.744	0.755	0.725	0.553	0.568	0.563	0.765	0.781	0.730	0.866	0.933	**0.943**
Total	0	0	0	0	0	0	0	1	0	1	6	13

Box plot diagrams comparisons between proposed bGOQRFA and bFA for all 21 datasets are summarized in Fig. 5.

4.2.4. Additional experiments

Moreover, with the goal of further validation, the bGOQRFA method was also tested on two more real-life datasets, first dealing with COVID-19, and second with microarray and IoT microcontroller data. The microcontroller dataset was chosen because it contains a large number of features, and it can give additional insight into the performances of the proposed algorithm.

The bGOQRFA method was first employed for relatively new COVID-19 dataset for predicting patient health and compared to other-state-of-the art methods, as it is presented in [58]. Utilized COVID-19 dataset consists of 15 features and can be retrieved from here. Details of this dataset can be seen from [58].

When compared with other approaches from [58], the bGOARFA established the best accuracy of 93.41% and the second best approach HLBDA (hyper learning binary DA) obtained accuracy of 92.21%. On the other hand, bFA and bFAQRLI obtained accuracy of 88.45% and 90.33%, respectively. Fig. 6 presents box plot comparison between bFA and bGOQRFA for COVID-19 dataset for objective (fitness) value.

To validate the performances of the proposed algorithm on the datasets with a large number of features, another experiment was conducted with the microcontroller microarray dataset SMK-CAN-187, that is available on https://jundongl.github.io/scikit-feature/datasets.html. This experiment was inspired by the experiments and results published by Sun et al. [59]. SMK-CAN-187 dataset from the microarray domain is specific because of a large number of features — 19,993, with 187 instances and two classes.

The conducted experiments have shown that the basic bFA obtained the classification error of 0.13258. The proposed bGOQRFA algorithm achieved superior performances by obtaining the classification error of 0.10824, while selecting on average 74.92 features. In comparison to other methods described in [59], the proposed bGOQRFA outperformed all of them significantly on this particular dataset.

5. Conclusion

Feature selection is a very important pre-processing method in machine learning and data science for reducing the dimensionality of

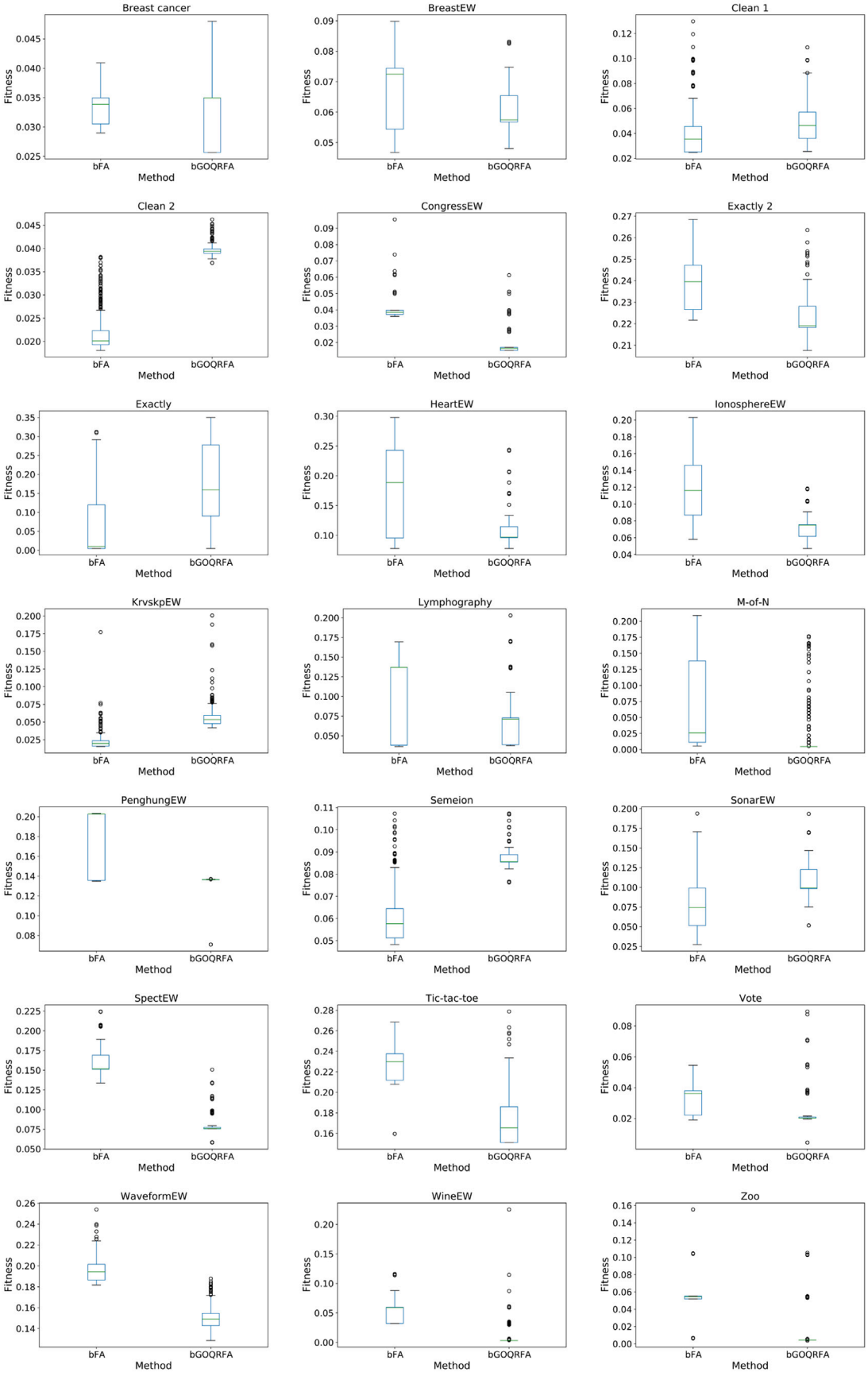


Fig. 5. Box plot comparison: bGOQRFA vs. bFA.

Table 14
Average number of selected features on 21 data sets for the compared algorithms.

No.	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA	bFA	bFAQRLI	bGOQRFA
1	6.0875	6.3875	5.6750	4.7500	5.0250	5.0875	6.3600	6.3875	5.0625	**3.7081**	4.572	8.075
2	7.7500	9.7083	7.5555	6.1806	6.3750	6.2083	5.2000	7.9167	8.0417	4.6440	5.1842	**5.139**
3	6.6172	7.6328	6.0625	6.2031	6.1797	6.2500	6.0900	5.9141	4.71090	5.6162	**3.3764**	5.0881
4	6.2596	6.9904	5.8365	5.5865	5.6154	5.4231	6.4300	5.8269	4.7019	4.6022	2.9510	**2.9251**
5	6.4602	7.3920	5.9148	5.4432	5.9886	5.6989	5.6800	6.2898	4.5966	6.1825	5.7640	**5.6100**
6	6.4729	6.6396	5.5667	6.0563	6.0566	6.2396	5.2000	6.2146	**4.3854**	10.3200	9.4805	11.0439
7	6.0221	6.6875	5.9265	5.4522	5.5699	5.4081	5.6400	6.1213	4.0625	9.5883	**3.6043**	9.1460
8	5.5577	5.4519	5.4134	5.1731	4.5769	4.7885	6.1100	5.7596	**4.1730**	5.0570	6.7732	6.6951
9	5.3281	5.8438	5.4609	5.0859	5.2578	5.0469	4.2700	6.2891	4.4219	5.6231	**3.6160**	8.3040
10	7.0417	9.0347	6.7951	6.1909	6.2535	6.2361	5.7800	7.6314	**5.3368**	15.2640	11.8442	15.4086
11	7.3344	9.0500	7.0750	6.2656	6.3156	6.3062	7.5000	7.9906	**5.8656**	18.7213	22.1245	10.4433
12	6.4038	7.2693	6.9712	5.1635	5.4231	5.4231	**4.7500**	6.2212	6.1827	6.8252	5.1489	9.4651
13	4.9904	4.6731	6.1538	3.9423	4.0385	4.4615	4.7500	4.2981	**1.7885**	5.8500	6.7860	9.1523
14	7.2404	8.7884	6.9135	6.2212	6.0865	6.2115	6.9500	7.6442	6.3462	5.0960	**2.7047**	3.9785
15	6.6719	7.4609	6.0234	5.9141	5.6640	6.1016	5.2300	6.1094	**3.7813**	5.7125	10.1124	5.5213
16	5.7250	6.2375	6.0250	5.1875	4.9500	5.1000	5.5200	6.0750	**4.8875**	15.8423	16.5392	21.725
17	6.6788	7.9953	5.9774	6.2183	6.2538	6.2363	8.5600	6.4108	**5.0028**	74.7329	17.4956	76.852
18	6.9247	7.9488	5.8893	6.2146	6.1942	6.2387	6.5700	6.4932	**4.8532**	47.64200	28.8842	44.6529
19	6.6822	7.7086	5.7515	6.2432	6.2402	6.2771	7.8100	6.8577	**4.8735**	55.2789	69.8860	122.0135
20	6.6250	7.2708	6.0069	6.0555	5.8958	5.9028	**4.9900**	6.2569	5.0486	14.4180	16.2542	13.9531
21	6.4835	7.1131	5.3630	6.2142	6.2111	6.2312	5.5000	4.9126	**4.7477**	108.5533	151.7752	181.0253
Total	0	0	0	0	0	0	2	0	11	1	4	3

Table 15
Friedman ranks of the comparable method over the 21 benchmark datasets.

Dataset	WOA	bWOA-S	bWOA-v	BALO1	BALO2	BALO3	PSO	bGWO	bDA	bFA	bFAQRLI	bGOQRFA
1	8	7	10	12	11	9	4	6	5	3	2	1
2	9	4	6.5	10	12	11	3	6.5	5	8	2	1
3	11	10	7	12	9	8	5	6	4	3	1	2
4	8	9	7	10	11.5	11.5	5	6	4	3	2	1
5	10	8	7	11	12	9	4	6	5	3	2	1
6	12	10	7	8	11	9	4	6	5	2	1	3
7	10	9	7	8	12	11	4	6	5	2	3	1
8	12	8.5	7	10	11	8.5	3	6	4	5	2	1
9	10	8	7	11	12	9	4	6	5	3	2	1
10	7	10	12	10	8	10	4	6	5	1	2.5	2.5
11	8	8	10.5	12	10.5	8	4	5	6	3	1	2
12	9	11.5	7.5	7.5	10	11.5	2	6	3	4	1	5
13	5	8	10.5	3	8	12	4	8	6	10.5	2	1
14	9	7	12	11	10	8	3	6	4	5	2	1
15	8.5	8.5	7	11	12	10	3	6	4	5	2	1
16	9	7	8	10.5	12	10.5	3.5	6	1.5	3.5	5	1.5
17	9.5	7	5	7	7	9.5	3	4	2	12	11	1
18	10	9	7	8	11	12	4	6	5	3	2	1
19	10.5	10.5	7	8	10.5	10.5	4	6	5	3	2	1
20	9	8	7	12	11	10	4	6	5	2	3	1
21	9	8	7	12	10	11	4	6	5	3	2	1
Average ranking	9.21	8.38	7.90	9.71	10.54	9.95	3.74	5.98	4.45	4.14	2.5	1.48
Rank	9	8	7	10	12	11	3	6	5	4	2	1

Table 16
Holm's step-down procedure result.

Comparison	*p*-value	Rank	0.05/(k-i)	0.1/(k-i)
bGOQRFA vs BALO2	2.22E−16	0	0.00455	0.00909
bGOQRFA vs BALO3	1.29E−14	1	0.00500	0.01000
bGOQRFA vs BALO1	6.62E−14	2	0.00556	0.01111
bGOQRFA vs WOA	1.77E−12	3	0.00625	0.01250
bGOQRFA vs bWOA-S	2.72E−10	4	0.00714	0.01429
bGOQRFA vs bWOA-v	3.79E−09	5	0.00833	0.01667
bGOQRFA vs bGWO	2.62E−05	6	0.01000	0.02000
bGOQRFA vs bDA	3.74E−03	7	0.01250	0.02500
bGOQRFA vs bFA	8.27E−03	8	0.01667	0.03333
bGOQRFA vs PSO	2.10E−02	9	0.02500	0.05000
bGOQRFA vs bFAQRLI	1.79E−01	10	0.05000	0.10000

data, which leads to computation time reduction and also can positively affect the classification accuracy. In this article, a novel approach is proposed for the feature selection problem by employing improved FA method, that enhances original implementation by utilizing GO and QRBL mechanisms.

Proposed method significantly improves performance of the basic FA, and also outperforms other state-of-the-art metaheuristics for both, benchmark bound-constrained and practical feature selection tasks. Method was first validated on 18 standard unconstrained benchmarks and later it was applied for feature selection by using 21 standard UCL

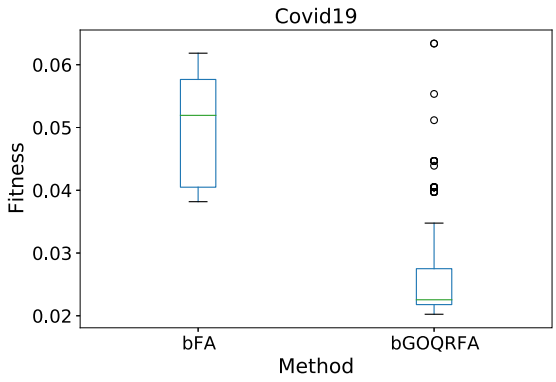


Fig. 6. Box plot comparison: bGOQRFA vs. bFA.

datasets , 1 relatively new COVID-19 dataset, and 1 microcontroller dataset. In all practical simulations proposed GOQRFA in average outscored all other metaheuristics in terms of convergence, solutions' quality and classification accuracy.

However, study presented in this manuscript also suffers from some limitations. Proposed GOQRFA is more complex, as it employs additional control parameters when compared to the basic FA method, which makes it harder to be adjusted by the researcher. Also, the potential of GOQRFA was not fully investigated on other real-world feature selection datasets. However, this will be subject of future research from this domain. Additionally, future research will also focus on testing the proposed GOQRFA algorithm on other NP-hard problems, including the task ordering in cloud-based systems, intrusion detection systems, and other challenges in the domain of artificial intelligence — feature selection and hyperparameter optimization of various neural network models.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by Ministry of Science of Republic of Serbia, Grant No. III-44006.

References

[1] S. Luo, L. Cheng, B. Ren, Practical swarm optimization based fault-tolerance algorithm for the Internet of Things, KSII Trans. Internet Inf. Syst. (TIIS) 8 (3) (2014) 735–748.

[2] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang, K. Long, Cognitive Internet of Things: A new paradigm beyond connection, IEEE Internet Things J. 1 (2) (2014) 129–143, http://dx.doi.org/10.1109/JIOT.2014.2311513.

[3] S. Messaoud, A. Bradai, S.H.R. Bukhari, P.T.A. Quang, O.B. Ahmed, M. Atri, A survey on machine learning in Internet of Things: Algorithms, strategies, and applications, Internet Things 12 (2020) 100314, http://dx.doi.org/10.1016/j.iot.2020.100314, URL https://www.sciencedirect.com/science/article/pii/S2542660520301451.

[4] X. Zenggang, Z. Mingyang, Z. Xuemin, Z. Sanyuan, X. Fang, Z. Xiaochao, W. Yunyun, L. Xiang, Social similarity routing algorithm based on socially aware networks in the big data environment, J. Signal Process. Syst. 94 (11) (2022) 1253–1267.

[5] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Comput. Electr. Eng. 40 (1) (2014) 16–28, http://dx.doi.org/10.1016/j.compeleceng.2013.11.024, URL https://www.sciencedirect.com/science/article/pii/S0045790613003066, 40th-year commemorative issue.

[6] H. Li, R. Peng, Z.-a. Wang, On a diffusive susceptible-infected-susceptible epidemic model with mass action mechanism and birth-death effect: analysis, simulations, and comparison with other mechanisms, SIAM J. Appl. Math. 78 (4) (2018) 2129–2153.

[7] B. Xue, M. Zhang, W.N. Browne, Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms, Appl. Soft Comput. 18 (2014) 261–276, http://dx.doi.org/10.1016/j.asoc.2013.09.018, URL https://www.sciencedirect.com/science/article/pii/S1568494613003128.

[8] R. Sun, J. Wang, Q. Cheng, Y. Mao, W.Y. Ochieng, A new IMU-aided multiple GNSS fault detection and exclusion algorithm for integrated navigation in urban environments, GPS Solut. 25 (2021) 1–17.

[9] M. Sharma, P. Kaur, A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem, Arch. Comput. Methods Eng. 28 (3) (2021).

[10] L.-Y. Chuang, S.-W. Tsai, C.-H. Yang, Improved binary particle swarm optimization using catfish effect for feature selection, Expert Syst. Appl. 38 (10) (2011) 12699–12707.

[11] C. Huang, Z. Han, M. Li, X. Wang, W. Zhao, Sentiment evolution with interaction levels in blended learning environments: Using learning analytics and epistemic network analysis, Australas. J. Educ. Technol. 37 (2) (2021) 81–95.

[12] A. Yan, Z. Li, J. Cui, Z. Huang, T. Ni, P. Girard, X. Wen, LDAVPM: a latch design and algorithm-based verification protected against multiple-node-upsets in harsh radiation environments, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2022).

[13] A. Yan, Z. Xu, K. Yang, J. Cui, Z. Huang, P. Girard, X. Wen, A novel low-cost TMR-without-voter based HIS-insensitive and MNU-tolerant latch design for aerospace applications, IEEE Trans. Aerosp. Electron. Syst. 56 (4) (2019) 2666–2676.

[14] T. Bezdan, D. Cvetnic, L. Gajic, M. Zivkovic, I. Strumberger, N. Bacanin, Feature selection by firefly algorithm with improved initialization strategy, in: 7th Conference on the Engineering of Computer Based Systems, 2021, pp. 1–8.

[15] K. Wang, B. Zhang, F. Alenezi, S. Li, Communication-efficient surrogate quantile regression for non-randomly distributed system, Inform. Sci. 588 (2022) 425–441.

[16] X. Xie, B. Xie, D. Xiong, M. Hou, J. Zuo, G. Wei, J. Chevallier, New theoretical ISM-K2 Bayesian network model for evaluating vaccination effectiveness, J. Ambient Intell. Humaniz. Comput. (2022) 1–17.

[17] D. Yang, T. Zhu, S. Wang, S. Wang, Z. Xiong, LFRSNet: a robust light field semantic segmentation network combining contextual and geometric features, Front. Environ. Sci. (2022) 1443.

[18] Z. Zhu, Y.-S. Ong, M. Dash, Wrapper–filter feature selection algorithm using a memetic framework, IEEE Trans. Syst. Man Cybern. B 37 (1) (2007) 70–76.

[19] O.H. Babatunde, L. Armstrong, J. Leng, D. Diepeveen, A genetic algorithm-based feature selection, IJECCE (2014).

[20] X. Li, X. Zhang, T. Jia, Humanization of nature: Testing the influences of urban park characteristics and psychological factors on collegers' perceived restoration, Urban For. Urban Green. 79 (2023) 127806.

[21] M. Zivkovic, N. Bacanin, T. Zivkovic, I. Strumberger, E. Tuba, M. Tuba, Enhanced grey wolf algorithm for energy efficient wireless sensor networks, in: 2020 Zooming Innovation in Consumer Technologies Conference, ZINC, IEEE, 2020, pp. 87–92.

[22] M. Zivkovic, N. Bacanin, K. Venkatachalam, A. Nayyar, A. Djordjevic, I. Strumberger, F. Al-Turjman, COVID-19 cases prediction by using hybrid machine learning and beetle antennae search approach, Sustainable Cities Soc. 66 (2021) 102669.

[23] M. Zivkovic, T. Bezdan, I. Strumberger, N. Bacanin, K. Venkatachalam, Improved Harris Hawks optimization algorithm for workflow scheduling challenge in cloud–edge environment, in: Computer Networks, Big Data and IoT, Springer, 2021, pp. 87–102.

[24] M. Dorigo, M. Birattari, Ant Colony Optimization, Springer, 2010.

[25] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, 1995, pp. 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968.

[26] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697.

[27] X.-S. Yang, A. Hossein Gandomi, Bat algorithm: a novel approach for global engineering optimization, Eng. Comput. 29 (5) (2012) 464–483.

[28] X.-S. Yang, Firefly algorithms for multimodal optimization, in: O. Watanabe, T. Zeugmann (Eds.), Stochastic Algorithms: Foundations and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 169–178.

[29] N. Khare, P. Devan, C.L. Chowdhary, S. Bhattacharya, G. Singh, S. Singh, B. Yoon, Smo-dnn: Spider monkey optimization and deep neural network hybrid classifier model for intrusion detection, Electronics 9 (4) (2020) 692.

[30] A.G. Hussien, M. Amin, A self-adaptive Harris Hawks optimization algorithm with opposition-based learning and chaotic local search strategy for global optimization and feature selection, Int. J. Mach. Learn. Cybern. (2021) 1–28.

[31] S.K. Baliarsingh, W. Ding, S. Vipsita, S. Bakshi, A memetic algorithm using emperor penguin and social engineering optimization for medical data classification, Appl. Soft Comput. 85 (2019) 105773.

[32] T. Bezdan, M. Zivkovic, N. Bacanin, A. Chhabra, M. Suresh, Feature selection by hybrid brain storm optimization algorithm for COVID-19 classification, J. Comput. Biol. (2022).

[33] M. Zivkovic, C. Stoean, A. Chhabra, N. Budimirovic, A. Petrovic, N. Bacanin, Novel improved salp swarm algorithm: An application for feature selection, Sensors 22 (5) (2022) 1711.

[34] I. Strumberger, A. Rakic, S. Stanojlovic, J. Arandjelovic, T. Bezdan, M. Zivkovic, N. Bacanin, Feature selection by hybrid Binary Ant Lion Optimizer with COVID-19 dataset, in: 2021 29th Telecommunications Forum, TELFOR, IEEE, 2021, pp. 1–4.
[35] N. Bacanin, A. Petrovic, M. Zivkovic, T. Bezdan, M. Antonijevic, Feature selection in machine learning by hybrid sine cosine metaheuristics, in: International Conference on Advances in Computing and Data Sciences, Springer, 2021, pp. 604–616.
[36] S.S. Kareem, R.R. Mostafa, F.A. Hashim, H.M. El-Bakry, An effective feature selection model using hybrid metaheuristic algorithms for IoT intrusion detection, Sensors 22 (4) (2022) 1396.
[37] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, M. Tuba, Monarch butterfly optimization based convolutional neural network design, Mathematics 8 (6) (2020) 936.
[38] S. Milosevic, T. Bezdan, M. Zivkovic, N. Bacanin, I. Strumberger, M. Tuba, Feed-forward neural network training by hybrid bat algorithm, in: Modelling and Development of Intelligent Systems: 7th International Conference, MDIS 2020, Sibiu, Romania, October 22–24, 2020, Revised Selected Papers 7, Springer International Publishing, 2021, pp. 52–66.
[39] J. Yang, L. Qu, Y. Shen, Y. Shi, S. Cheng, J. Zhao, X. Shen, Swarm intelligence in data science: Applications, opportunities and challenges, in: International Conference on Swarm Intelligence, Springer, 2020, pp. 3–14.
[40] X.-S. Yang, H. Xingshi, Firefly algorithm: Recent advances and applications, Int. J. Swarm Intell. 1 (1) (2013) 36–50.
[41] I. Strumberger, E. Tuba, N. Bacanin, M. Zivkovic, M. Beko, M. Tuba, Designing convolutional neural network architecture by the firefly algorithm, in: Proceedings of the 2019 International Young Engineers Forum, YEF-ECE, Costa Da Caparica, Portugal, 2019, pp. 59–65.
[42] N. Bacanin, M. Tuba, Firefly algorithm for cardinality constrained mean-variance portfolio optimization problem with entropy diversity constraint, The Scientific World Journal, Special Issue Computational Intelligence and Metaheuristic Algorithms with Applications 2014 (2014) 16, http://dx.doi.org/10.1155/2014/721521, Article ID 721521.
[43] I. Strumberger, N. Bacanin, M. Tuba, Enhanced firefly algorithm for constrained numerical optimization, IEEE congress on evolutionary computation, in: Proceedings of the IEEE International Congress on Evolutionary Computation, CEC 2017, 2017.
[44] H. Wang, X. Zhou, H. Sun, X. Yu, J. Zhao, H. Zhang, L. Cui, Firefly algorithm with adaptive control parameters, Soft Comput. 3 (2017) 5091—5102.
[45] Q. Fan, Z. Chen, Z. Xia, A novel quasi-reflected Harris Hawks optimization algorithm for global optimization problems, Soft Comput. (2020) 1–19.
[46] R. Cazacu, Comparative study between the improved implementation of 3 classic mutation operators for genetic algorithms, Procedia Eng. 181 (2017) 634–640, http://dx.doi.org/10.1016/j.proeng.2017.02.444, URL http://www.sciencedirect.com/science/article/pii/S1877705817310287, 10th International Conference Interdisciplinarity in Engineering, INTER-ENG 2016, 6-7 October 2016, Tirgu Mures, Romania.
[47] T. Bäck, H. Schwefel, An overview of evolutionary algorithms for parameter optimization, Evol. Comput. 1 (1) (1993) 1–23.
[48] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Quasi-oppositional differential evolution, in: 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 2229–2236.
[49] M. Tuba, N. Bacanin, Improved seeker optimization algorithm hybridized with firefly algorithm for constrained optimization problems, Neurocomputing 143 (2014) 197–207, http://dx.doi.org/10.1016/j.neucom.2014.06.006.
[50] J. Liu, Y. Mao, X. Liu, Y. Li, A dynamic adaptive firefly algorithm with globally orientation, Math. Comput. Simulation 174 (2020) 76–101, http://dx.doi.org/10.1016/j.matcom.2020.02.020, URL http://www.sciencedirect.com/science/article/pii/S0378475420300598.
[51] Q. Zhu, Y. Xiao, W. Chen, C. Ni, Y. Chen, Research on the improved mobile robot localization approach based on firefly algorithm, 2016, pp. 323–329, 37.
[52] A. Kaveh, S. Javadi, Chaos-based firefly algorithms for optimization of cyclically large-size braced steel domes with multiple frequency constraints, Comput. Struct. 214 (2019) 28–39, http://dx.doi.org/10.1016/j.compstruc.2019.01.006, URL http://www.sciencedirect.com/science/article/pii/S0045794918314755.
[53] X.-S. Yang, Firefly algorithm, Lévy flights and global optimization, in: M. Bramer, R. Ellis, M. Petridis (Eds.), Research and Development in Intelligent Systems XXVI, Springer London, London, 2010, pp. 209–218.
[54] S. Yu, S. Zhu, Y. Ma, D. Mao, A variable step size firefly algorithm for numerical optimization, Appl. Math. Comput. 263 (2015) 214–220, http://dx.doi.org/10.1016/j.amc.2015.04.065, URL http://www.sciencedirect.com/science/article/pii/S0096300315005275.
[55] K.K. Ghosh, R. Guha, S.K. Bera, N. Kumar, R. Sarkar, S-shaped versus V-shaped transfer functions for binary Manta ray foraging optimization in feature selection problem, Neural Comput. Appl. (2021) 1–15.
[56] D. Dua, C. Graff, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2019.
[57] A.G. Hussien, D. Oliva, E.H. Houssein, A.A. Juan, X. Yu, Binary whale optimization algorithm for dimensionality reduction, Mathematics 8 (10) (2020) http://dx.doi.org/10.3390/math8101821, URL https://www.mdpi.com/2227-7390/8/10/1821.
[58] J. Too, S. Mirjalili, A hyper learning binary dragonfly algorithm for feature selection: A COVID-19 case study, Knowl.-Based Syst. 212 (2021) 106553.
[59] G. Sun, J. Li, J. Dai, Z. Song, F. Lang, Feature selection for IoT based on maximal information coefficient, Future Gener. Comput. Syst. 89 (2018) 606–616.

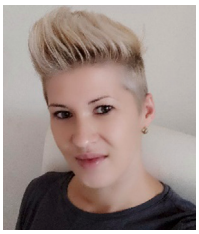
Nebojsa Bacanin received his Ph.D. degree from Faculty of Mathematics, University of Belgrade in 2015 (study program Computer Science, average grade 10,00). He started University career in Serbia 13 years ago at Graduate School of Computer Science in Belgrade. He currently works as an associate professor and as a vice-dean at Faculty of Informatics and Computing, Singidunum University, Belgrade, Serbia.

He is involved in scientific research in the field of computer science and his specialty includes stochastic optimization algorithms, swarm intelligence, soft-computing and optimization and modeling, as well as artificial intelligence algorithms, swarm intelligence, machine learning, image processing and cloud and distributed computing. He has published more than 120 scientific papers in high quality journals and international conferences indexed in Clarivate Analytics JCR, Scopus, WoS, IEEExplore, and other scientific databases, as well as in Springer Lecture Notes in Computer Science and Procedia Computer Science book chapters. He has also published 2 books in domains of Cloud Computing and Advanced Java Spring Programming.

He is a member of numerous editorial boards, scientific and advisory committees of international conferences and journals. He is a regular reviewer for international journals with high Clarivate Analytics and WoS impact factor such as Journal of Ambient Intelligence & Humanized Computing, Soft Computing, Applied Soft Computing, Information Sciences, Journal of Cloud Computing, IEEE Transactions on Computers, IEEE Review, Swarm and Evolutionary Computation, Journal of King Saud University C Computer and Information Sciences, SoftwareX, Neurocomputing, Operations Research Perspectives, etc. He actively participates in 1 national and 1 international projects from the domain of computer science. He has also been included in the prestigious Stanford University list with 2% best world researchers for the year 2020.

Dr K. Venkatachalam received his Bachelors in Information Technology in 2005, Master's in Computer Science and Engineering in 2008 and Ph.D. in Computer Science and Engineering in 2018. He has more than 13 years of academic experience and currently working as an Assistant Professor (SR) in the School of computer science and Engineering at VIT Bhopal University, Bhopal, Madhya Pradesh, and India. He had published several articles in peer-reviewed journals and his research interest includes Data Mining, Web services, semantic web services, Distributed computing, Cloud Computing.

He is a Sun Certified SCJP professional and has obtained Brain Bench certification in various disciplines. He has organized several workshops on J2ME, Advanced Java Programming, Web Services, Enterprise Computing, Web Technology and WSN (wireless sensor network) in his Institution and has presented papers in web services at National and International conferences. He has guided a number of research-oriented as well as application oriented projects organized by well known companies like IBM. He has delivered more than 20 Guest Lectures in reputed engineering colleges on various topics.



Timea Bezdan is working as Teaching Assistant at University Singidunum in Belgrade, Serbia at the Technical Faculty and Faculty of Informatics and Computing. She received B.S. degree in Software and Data Engineering from Singidunum University, Belgrade in 2020. She is currently pursuing M.S. degree with Contemporary Information Technologies, Singidunum University, Belgrade. Her research interest includes artificial intelligence, machine learning, optimization, swarm intelligence, and cloud computing. She has published more than 10 scientific papers in high-quality journals and international conferences in these areas.

Miodrag Zivkovic received his Ph.D. degree from School of Electrical Engineering, University of Belgrade in 2014 (study program Software Engineering, average grade 9,90). He started University career in Serbia 4 years ago at Singidunum University in Belgrade. He currently works as an assistant professor at Faculty of Informatics and Computing, Singidunum University, Belgrade, Serbia.

He is involved in scientific research in the field of computer science and his specialty includes object-oriented programming, mobile applications programming, software testing, stochastic optimization algorithms, swarm intelligence, human–computer interaction, as well as artificial intelligence algorithms. He has published more than 20 scientific papers in high quality journals and international conferences indexed in Clarivate Analytics JCR, Scopus, WoS, IEEExplore, and other scientific databases, as well as in Springer Lecture Notes in Computer Science. He has also published 3 books in domains of Software Testing, Mobile Applications Development, and Advanced Java Spring Programming.

He is a regular reviewer for international journals with high Clarivate Analytics and WoS impact factor such as Computer Applications in Engineering Education, IEEE Access, Journal of Intelligent Systems, Journal of Educational Computing Research, etc.

Dr. Mohamed Abouhawwash received the master's and Ph.D. degrees in statistics and computer science from Mansoura University, Mansoura, Egypt, in 2011 and 2015, respectively. He is a postdoctoral research associate at Computational Mathematics, Science, and Engineering (CMSE), Biomedical Engineering (BME) and Radiology, Institute for Quantitative Health Science & Engineering (IQ), Michigan State University, East Lansing, MI 48824, USA. He is an Assistant Professor with the Department of Mathematics, Faculty of Science, Mansoura University, Egypt. His current research interests include evolutionary algorithms, machine learning, image reconstruction, and mathematical optimization. Dr. Abouhawwash was a recipient of the best master's and Ph.D. thesis awards from Mansoura University in 2012 and 2018, respectively.