Complex & Intelligent Systems (2025) 11:141
https://doi.org/10.1007/s40747-024-01763-y

ORIGINAL ARTICLE

A novel group-based framework for nature-inspired optimization algorithms with adaptive movement behavior

Adam Robson[1,2] · Kamlesh Mistry[2] · Wai-Lok Woo[2]

Received: 24 May 2024 / Accepted: 20 December 2024

Abstract
This paper proposes two novel group-based frameworks that can be implemented into almost any nature-inspired optimization algorithm. The proposed Group-Based (GB) and Cross Group-Based (XGB) framework implements a strategy which modifies the attraction and movement behaviors of base nature-inspired optimization algorithms and a mechanism that creates a continuing variance within population groupings, while attempting to maintain levels of computational simplicity that have helped nature-inspired optimization algorithms gain notoriety within the field of feature selection. Through this functionality, the proposed framework seeks to increase search diversity within the population swarm to address issues such as premature convergence, and oscillations within the swarm. The proposed frameworks have shown promising results when implemented into the Bat algorithm (BA), Firefly algorithm (FA), and Particle Swarm Optimization algorithm (PSO), all of which are popular when applied to the field of feature selection, and have been shown to perform well in a variety of domains, gaining notoriety due to their powerful search capabilities.

Keywords Classification · Feature selection · Nature-inspired algorithms · Optimization

Introduction

This paper presents two frameworks designed to address issues within nature-inspired optimization algorithms and improve overall performance of the algorithms. This is achieved through the incorporation of a novel approach to group-based swarm dynamics, creating frameworks that can be implemented into almost any nature-inspired optimization algorithm that utilizes an attraction style mechanism to control movement. The first framework is the Group-Based (GB) framework, which formalizes the approach originally proposed in [18], and the second framework is the Cross Group-Based (XGB) framework. Both frameworks are briefly introduced in this section, and fully defined in Sect. 3. Firstly, this study formalizes the Group-Based (GB) framework, originally proposed in [18], which presented an initial implementation of the GB framework, creating the Group-Based Firefly Algorithm (GBFA). The GBFA showed promising results when benchmarked against eight well-known optimization problems (Ackley, Easom, Griewank, Michalewicz, Rastrigin, Rosenbrock, Schwefel and Sphere), outperforming the standard FA implementation, along with other recent studies relating to nature-inspired algorithms, suggesting that further research into this group-based swarm dynamic would be useful. Although nature-inspired swarm intelligence algorithms have proven themselves to be powerful and effective optimization techniques, they are still susceptible to issues such as premature convergence and oscillations within the swarms, both of which can negatively impact performance of the algorithms and cause swarm stagnation in sub-optimal domains [3]. The group-based augmentation addresses the aforementioned stagnation and oscillation issues by increasing the search diversity of swarms, through having them move in ways that would not be natural to the typical behaviors of the swarms. After the success of the initial GBFA implementation, a further framework was developed, the Cross Group-Based (XGB) framework, which allows cross-collaboration between the population groups. The main contributions of

✉ Adam Robson
adam.robson@sunderland.ac.uk

1 School of Computer Science and Engineering, Faculty of Business and Technology, University of Sunderland, Sunderland, U.K.

2 Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K.

the novel frameworks and algorithm variants proposed in this study can be noted as follows:

1) Within the proposed frameworks, both group-based approaches are shown to outperform the base implementations of the augmented algorithm.
2) The implementation of the group-based approaches within BA, FA and PSO has led to the creation of six augmented algorithms, all of which showed excellent performance when compared to other current feature selection algorithms.
3) The proposed frameworks have shown improved algorithm stability in different dimensionalities.
4) The proposed framework can be implemented into other nature-inspired optimization algorithms, as it does not rely on any specific underlying mechanics of BA, FA or PSO.
5) The GBFA has shown a lower complexity than the standard FA implementation.
6) The PSO algorithm augmented with the group-based approaches demonstrated the best overall performance of the augmented algorithms.

Nature-inspired optimization algorithms have shown successful application when used for feature selection in classification problems, and has shown success across a variety of domains such as facial expression recognition [14], data mining [8] and intrusion detection [7]. Feature selection is a crucial pre-processing method and is an imperative part of creating optimal machine learning models, particularly in datasets with high-dimensionality, which are often encountered in real-world applications. These datasets commonly contain redundant or irrelevant features, which can negatively affect the performance of machine learning algorithms, particularly in the learning process [1]. Robust and effective feature selection helps identify the most significant features to be used, and removes irrelevant and redundant features [9], thus reducing the dimensionality and ensuring that the most significant features are used. There are three main categories of feature selection methods, these are wrapper-based methods, filter-based methods and embedded methods [9]. Wrapper-based methods use machine learning techniques to select the most optimal feature subset, and regardless of higher computational cost, they are typically noted to provide a higher classification accuracy [3]. While in contrast filter-based methods, use statistical measurements to rank and select features independently of any learning algorithm, subsequently making them less computationally demanding, but sometimes less accurate. Datasets with high dimensionality can be problematic for classification problems within machine learning due to memory usage and high computational costs. Traditional optimization methods have been applied to feature selection problems, but suffer from performance issues such as premature convergence, or inconsistent performance across datasets with different dimensionalities [20]. Metaheuristic algorithms are widely considered to be the most useful and efficient methods for tackling datasets with high dimensionality, due to the large search space of features. Nature-inspired metaheuristics, such as evolutionary algorithms and swarm intelligence algorithms have seen particularly successful application within the area of dimensionality reduction, consistently outperforming traditional optimization methods [23].

Swarm intelligence algorithms are inspired by the collective behaviors of social swarms that occur within nature, modelling the exploration and exploitation behaviors of these social swarms. Swarm intelligence algorithms have been utilized successfully as wrapper methods for feature selection problems. The swarms consist of artificial agents, typically made up of a collection of unsophisticated agents, that demonstrate a coordinated behavior to achieve the desired goal of the swarm. Agents within the swarm interact with each other, creating a self-organizing and decentralized swarm. Swarm intelligence metaheuristic algorithms such as Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), Bat Algorithm (BA), Firefly Algorithm (FA) and Particle Swarm Optimization (PSO), have been effectively used as efficient and robust optimizers for a wide range of NP-hard problems across a variety of different domains [3, 4, 8, 11, 20, 23]. Modified versions of nature-inspired metaheuristic algorithms have been proposed in previous research, and generally make modifications or augmentations that focus upon manipulation of parameter values, search strategies, or creating hybrid combinations. Group-based modifications have also been previously proposed, with variations in functionality and grouping paradigms implemented, but these often heavily impact the complexity of the algorithm, which can be seen in [28]. Current group-based strategies also rely predominantly on the underlying behavior of the modified algorithm, making these approaches difficult to adapt to other algorithms and often meaning that parameters must be adjusted to offer the same level of performance in different dimensionalities or problem domains [5, 22, 28]. For example, group-based strategies have been implemented into FA in research such as [5], which implements a grouping behavior based on an additional visual field and a global best, with fireflies grouping with those visible to it. An additional example of a group-based FA can be seen in [22], which implements an elite group, that exchanges information with sub-groups, and also allows individual fireflies to jump out of their current sub-group to a location near the highest performing fireflies. The PSO algorithm has also seen grouping mechanisms implemented, for example in [6], a hybrid algorithm which utilizes the PSO algorithm and the Crow Search (CS) algorithm. Within this research, groups are formed at the initialization stage, and then the best performers from each group are distributed into other groups as the algorithm continues to search until the termination criteria is met. The aforementioned group-based modifications have seen some successes and show that implementing group-based paradigms into nature-inspired metaheuristic algorithms can be useful in improving performance.

The rest of the paper is organized as follows: The Related Work section contains a review of related literature, highlighting the key theories, concepts and areas this paper seeks to address. The Proposed Frameworks section contains an explanation of the functionality of the proposed frameworks, an overview of the experimentation system design, and the datasets used. The Results section presents data analysis and findings of this study, and an evaluation of each of the algorithm variants derived from the proposed framework. Finally, the Conclusion section summarizes and discusses the main findings and their implications, and makes recommendations for future research.

Related work

Bat algorithm

The Bat Algorithm (BA) was originally proposed by [26] as a metaheuristic optimization technique based upon the echolocation communication and movement behaviors of a group of bats when tracking food or prey. BA has been applied in various optimization problems and has shown successes, as noted in [20, 23]. Echolocation functions as a form of sonar, with bats emitting a short and loud pulse of sound. The bats wait for the echo to return to them after it hits an obstacle or object, using the time delay to allow calculate their distance from the obstacle or object. The echolocation used by bats is not purely for navigation and they also have the ability to tell the difference between an obstacle or object, and food or prey they are hunting. The algorithm proposed by [26], has the following idealized rules:

1) All bats use echolocation to sense distance, and they also know the difference between food/prey and background barriers;
2) When searching for food or prey, bats (x_i) fly randomly with a velocity (v_i) at a position (x_i), with a fixed frequency (f_{min}), and each bat also has a varying wavelength (λ) and loudness (A_0). Bats can automatically adjust the wavelength (or frequency) of the pulses they emit, and can also adjust the rate of pulse emission $r \in [0,1]$, depending on the proximity of their target;
3) Although the loudness can vary in many ways, it is assumed that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

The standard BA begins with the initialization of a swarm of virtual bats (x_i), with each bat assigned a velocity (v_i), with the frequencies (f_i), pulse emission rates (r_i) and the loudness (A_i) also initialised. Initially, each bat (x_i) is given a random frequency (f_i), drawn uniformly from $[f_{min}, f_{max}]$. The algorithm will then repeat the movement and search strategy until the maximum number of iterations has been reached, with t acting as the iteration counter.

The movement of bats is controlled by rules that determine how their positions (x_i) and velocities (v_i) in a d -dimensional search space are updated, with the new solutions (x_i^t) and new velocities (v_i^t) at iteration t given by the equations shown in (1), (2) and (3), where $\beta \in [0,1]$ is a random vector drawn from a uniform distribution and the current global best solution (location) is represented by x_* , which is determined after comparing all solutions among the swarm of bats (x_i).

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{1}$$

$$v_i^{t+1} = v_i^t + \left(x_i^t - x_*\right) f_i \tag{2}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{3}$$

The product of $\lambda_i f_i$ is a constant, therefore we can use f_i (or λ_i) to adjust velocity change, while fixing the other factor λ_i (or f_i), depending on the problem type. In the standard implementation of BA, $f_{min} = 0$ is used, with $f_{max} = O(1)$, depending on the size of the search domain. In terms of local search, after a global best has been assigned, a new solution for each bat (x_i) is generated through the use of a local random walk, as shown in (4), with ϵ is a random number in $\in [-1,1]$ and $A^t = A_i^t$ represents the average loudness of all bats in this iteration.

$$x_{new} = x_{old} + \epsilon A^t \tag{4}$$

A scaling parameter can also be used to control the step size, as shown in (5), where ϵ_t is assigned from a Gaussian normal distribution $N(0,1)$ and a scaling factor is present with σ . The scaling factor should be linked to the scalings of the design variables of the optimisation problem.

$$x_{new} = x_{old} + \sigma\, \epsilon_t A^t \tag{5}$$

The velocity and position update of the bats is similar to that seen in the PSO algorithm, discussed later in this section, as the f_i essentially controls the range of movement and velocity of swarming particles. However, it is noted that BA can prove more effective in some applications, as it utilises parameter control and frequency tuning to influence the exploitation and exploration [26]. As the iterations proceed, the loudness (A_i) and pulse emission rate (r_i) are also updated as appropriate. In standard functionality, the loudness usually decreases when a bat (x_i) is approaching the best solution, with the rate of decrease determined by α in (6).

$$A_i^{t+1} = \alpha \, A_i^t \tag{6}$$

The loudness value is noted to play an important part in obtaining good quality solutions with the BA [21], and the configuration of the minimum loudness (A_{min}) and maximum loudness (A_{max}) depends on the domain application and the size of the dataset. The pulse emission rate (r_i) determines whether a local search around the global best solution should be performed or skipped. As the bat (x_i) approaches the global best, the pulse rate value will increase, subsequently decreasing the likelihood of the bat conducting a local search, as the bat is less likely to perform a local search around the global best as the pulse rate increases. Pulse rate increase is controlled by the equation shown in (7) and is determined by γ .

$$r_i^{t+1} = r_i^0 [1 - \exp{(-\gamma \, t)}] \tag{7}$$

Firefly algorithm

Firefly Algorithm (FA) was developed originally in 2008 by Yang and is a relatively new nature-inspired metaheuristic algorithm, with applications and additional advances shown in [18]. FA has been successfully applied to a variety of optimization problems [2–4]. The FA is a stochastic search algorithm, based around a population and has similar functionality to the PSO algorithm, discussed later in this section. The pseudocode for the standard FA can be seen in Algorithm 2. Since its initial conception, FA has gained notoriety for the powerful search capability it offers and overall computational simplicity. The algorithmic design concepts of the FA are based upon the luminescence attribute of tropical fireflies, which influences the behaviors and movements of a swarm [10], creating a virtual swarm consisting of autonomous agents, which is self-organizing and decentralized. The standard implementation of FA is also based on the following idealized rules [27]:

1) All fireflies within the swarm are unisex and therefore all fireflies will be attracted to one and other regardless of gender;
2) The attractiveness of a firefly is proportional to the brightness, with brightness decreasing as distance increases;
3) For any two flashing fireflies, the less bright of the two will move toward the brighter one;
4) If there is no brighter firefly, it will move randomly within the search domain;
5) The brightness of an individual firefly is determined by the objective function ($f\,(x)$).

There are two important factors which influence the FA: the formulation of attractiveness and the light intensity variation. Adjustment of these two factors allows developers to fine tune the firefly algorithm in a manner that is best suited to the requirements of the problem being solved. The attractiveness of a firefly is determined by the brightness attribute (I), which is proportional to the objective function ($f\,(x)$). Therefore, the brightness (I) of a firefly at a location (x) within the search domain can be determined by $I\,(x) \propto \; f\,(x)$. The attractiveness (β) of a firefly is relative and, as per the idealised rules of FA, should be determined by other fireflies within the swarm. The brightness intensity will vary based upon the distance (r_{ij}) between firefly i and firefly j, with the light intensity of a firefly decreasing with distance from the source. Additionally, a level of absorption is also considered within the FA, to allow for light absorbed within the search domain. This means that the light intensity ($I\,(r)$), in its simplest form, varies according to the inverse square law, shown in (8), with the light intensity at source is donated as I_s.

$$I\,(r) = \frac{I_s}{r^2} \tag{8}$$

The light intensity can then be determined by the equation shown in (9), with a fixed light absorption coefficient (γ), and the light intensity of the source denoted as I_0 at distance $r = 0$.

$$I\,(r) = I_0 e^{-\gamma \, r^2} \tag{9}$$

Additionally, within Eq. (9), the singularity at $r = 0$ within the expression I_s/r^2 is avoided by the combined effects of the approximation of absorption in Gaussian form, and inverse square law. As the attractiveness of a firefly is directly proportional to the intensity of the brightness, the attractiveness (β) of a firefly can be defined as the equation shown in (10), with the attractiveness at $r = 0$ represented by $\beta_{\,0}$.

$$\beta = \beta_0 e^{-yr^2} \tag{10}$$

The Euclidean distance between two fireflies, x_i and x_j, can be expressed as shown in (11), where the dimensionality of the problem is denoted by n.

$$r_{ij} = || x_i - x_j || = \sqrt{\sum{}_{k=1}^{k=n} (x_{i,k} - x_{j,k})^2} \tag{11}$$

Firefly movement of firefly x_i is based on the level of attraction to firefly x_j, and can be expressed as shown in (12),

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2} \left(x_j^t - x_i^t\right) + \alpha \epsilon_i^t \tag{12}$$

where firefly x_i will move toward firefly x_j if it has a higher intensity of brightness. The first term of (12) represents the current location of firefly x_i, the second term representing the movement from one position to another based on attraction and finally a random walk consisting of a randomization parameter (α), and ϵ_i is a vector of random numbers drawn from a Gaussian distribution with the interval $[0, 1]$. In the event that $\beta_0 = 0$, the firefly will take a simple random walk.

Particle swarm optimization

The PSO algorithm was originally developed by Kennedy and Eberhart in 1995 and has become a heavily used optimization algorithm, across a variety of problem domains [3, 11, 19], due to its computational simplicity, flexibility and powerful search capability. PSO is inspired by the collaborative and swarming behaviors seen in biological populations such as schools of fish, or flocks of birds. A standout feature of the PSO algorithm, compared to other swarm-intelligence-based algorithms, is that it uses global communication among the swarm, and real-number randomness. As noted in [26], the functionality of some components within the PSO have given inspiration to new swarm intelligence algorithms, effectively pioneering the basic ideas of swarm-intelligence-based computation, where knowledge is optimized by social interaction within the population and thinking can be considered both personal and social.

PSO algorithm uses a swarm of individual particles (agents) and performs searches of the problem domain via an objective function ($f(x)$), which is used to adjust the trajectory of individual particles within the swarm. The movement of the individual particles is determined and controlled by two components: deterministic component, and a stochastic component. Within each swarm, there exists a global best (g^*), which represents the current best solution found by the swarm. The movement of each particle within the swarm is based upon a velocity attribute (v_i), which is determined by attraction toward the position of the global best particle, and the best solution found by that particle within its history, the personal best (x_i^*).

Individual particles also have a tendency to move randomly within the search space. All particles within the swarm have a personal best attribute, and as the individual particles search within the problem domain, when a solution that is better than any previously found, the personal best is updated for that particle. The swarm will continue to search for a set number of iterations (t), or until a termination criterion is hit. To further expand upon the movement of individual particles, let x_i represent the position and v_i represent the velocity for particle i, the velocity vector is determined by the equation shown in (13).

$$v_i^{t+1} = v_i^t + \alpha \epsilon_1 \left[g^* - x_i^t\right] + \beta \epsilon_2 [x_i^{*(t)} - x_i^t] \tag{13}$$

In (13), ϵ_1 and ϵ_2 are two random vectors between $\in [0,1]$ and the parameters α and β are the learning parameters or acceleration constants. In the initialisation of the swarm, particles should be distributed in a relatively uniform manner, to allow coverage across most regions with the search domain, which is imperative for multimodal problems. When first initialised, the velocity of a particle can be taken as zero and represented as $v_i^{t=0} = 0$, with the position then being updated by the equation shown in (14), with the velocity (v_i) bounded within a defined range $\in [0, v_{max}]$.

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{14}$$

Feature selection in classification

Datasets with high-dimensional data are becoming more common place in the field of machine learning and can cause issues that lead to a degradation in performance, such as overfitting [12]. Datasets consisting of high-dimensional data can also increase computational costs required to perform data analytics, and the memory storage requirements. The primary goal of feature selection is to reduce the dimensionality of datasets by selecting the best and most informative features for classification problems, while also eliminating the redundant features. Feature selection as a pre-processing technique can improve the speed and accuracy of classification, while also reducing computation time, and the memory requirements for data storage. A dataset can be defined as $s = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \cong [x_{i1}, x_{i2}, \ldots, x_{id}]$ is a multi-dimensional vector sample, with n denoting the number of samples, d representing the number of features, and $y_i \in \gamma$ denotes the label of sample x_i [16]. In classification systems, choosing an optimization method to reduce the dimensionality of the dataset is vitally important to overall efficiency and performance, and nature-inspired metaheuristic algorithms such as BA, FA and PSO have shown to offer this capability [2–4, 6, 19].

k-nearest-neighbor classifier

The k-Nearest-Neighbor (k-NN) algorithm, introduced by Fix and Hodges in 1951, is a simple supervised machine learning algorithm and has seen widespread usage as a classifier in pattern recognition problems [15, 25]. The k-NN algorithm is non-parametric, which means that the data passed to the classifier does not need to conform to a normal distribution. Training data is inputted into a k-NN algorithm, and class membership is outputted. Objects are classified through majority votes of the k closest neighbours according to a similarity or distance function, where k is user defined constant in the form of a positive and typically small integer. The training data provided to the classifier consists of vectors in a multi-dimensional feature space, each of which is associated with a class label. As it is an instance-based learning algorithm, it does not explicitly construct an internal model, simply memorising the training dataset provided and uses this as "knowledge" in the prediction phase at the time of classification.

Related approaches

Group-based augmentations to nature-inspired optimization algorithms have been implemented with successful boosts in performance, as can be seen in multi-swarm approach studies such as [5, 22, 28]. However, these approaches have issues such as the additional overheads of group management and the additional computation required, along with issues related to adjusting the parameters for these algorithms to work in different problem domains, such as datasets with a low or high number of features, unimodal, or multimodal problems. Within the work of [28] for example, a modified Moth-Flame Optimization Algorithm (MFO) is proposed, the Multi-swarm Improved Moth–Flame Algorithm (MIMFO). While the proposed algorithm variant does increase population diversity and performance, the complexity of the algorithm has increased significantly, through the inclusion of a complex re-grouping methodology, a gaussian mutation, and also, the incorporation of additional search strategies. The additional augmented features of the MIMFO algorithm also make this difficult to translate to other nature-inspired algorithms, as they rely heavily on the functionality of the underlying MFO.

In other similar work, Cao et al. present the Visual Firefly Algorithm (VFA), which incorporates a multi-group approach which utilizes a visual field and relies on information exchange between groups, which, as noted within the results of their work, shows an expensive computational cost. The VFA also shows a lack of robustness in different problem domains, as it shows only a consistently improved performance on some multimodal problems, and has also been shown to reduce search diversity in some problems. Previous work such as [5, 18, 22], have shown the positive results that can be achieved through the incorporation of grouping behaviors into nature-inspired algorithms. A Multi-Group Firefly Algorithm based on Improved Evolutionism (IMGFA), is presented in [22] and shows some good results when compared to the standard implementation of other nature-inspired algorithms. The IMGFA uses an elite group that is constructed of the best performing firefly agents, which are then dispersed at each iteration into the poorly performing sub-groups, in an attempt to increase performance. If a sub-group cannot be improved by an elite group agent, then a mutation is applied to the best performing agent of the group. While the IMGFA has shown that groupings are capable of improving performance, the additional group management and mutations at each iteration have reduced the computational simplicity of the algorithm. In other work, a group-based mechanism is implemented into FA that assigns groupings based upon visual fields and an observer strategy, the VFA, is proposed in [5]. While improved results were noted with the VFA, the implied overhead of the additional behaviors added to the algorithmic design of the standard FA, along with the addition of the visual field component, the computational complexity of the algorithm is drastically increased, and again, poses issues translating these modifications into other nature-inspired optimization algorithms. While it is important to increase the search diversity of the swarm, increases in computational simplicity can have negative impacts on real-time systems implementing nature-inspired swarm intelligence algorithms, especially in areas that require updating to changes in the problem domain, such as intrusion detection [7].

In summary, current work within the field shows a lack of robustness and consistency when applied to different problem domains, and often performs well in certain applications but poorly in other areas. Additionally, the multi-swarm or group-based augmentations in current research are often implemented in a methodology specific to a single algorithm. This therefore means that the underlying paradigm and functionality of the algorithm augmentation is not easily transferrable to other nature-inspired algorithms, as they rely on specific functionality of the modified base algorithms. The GB and XGB frameworks proposed within this paper, however, do not rely on the movement behaviors or mechanisms of one algorithm, and can be implemented into almost any nature-inspired optimization algorithm.

Proposed frameworks

This paper proposes two novel frameworks that modify the behavior of nature-inspired optimization algorithms, the Group-Based (GB) framework and the Cross Group-based (XGB) framework, both of which seek to address common issues within nature-inspired swarm intelligence algorithms: premature convergence and oscillations within swarms, which can both lead to stagnation in sup-optimal domains. Oscillations within swarms are generally caused by either too many or too few attractions during the search process, which makes the mechanic of attraction, and ultimately movement, of individual agents, an important area of research. These issues are addressed in the proposed framework by increasing the search diversity of the swarm population, by having the agents act and move in a different way to their standard behavior, while attempting to maintain as much of the standard algorithm implementation, computational simplicity, and powerful search capability of the nature-inspired algorithm.

Group-based framework

In standard implementations of nature-inspired algorithms, the agents will move in a way determined by the rules of that algorithm, for example, in the standard FA, the firefly will instinctively move toward the most intensely bright firefly that is in closest proximity to them. In the GB approach, groupings are dynamically allocated at each iteration of the algorithm, with a group leader (x_{gl}) assigned, and agents within each group will move toward their group leader if it demonstrates a better performance when evaluated using the objective function ($f\left(x\right)$). However, if the group leader demonstrates a worse performance when evaluated using the objective function, the agent will continue to move within the search space based upon the standard functionality of that algorithm. While the groupings are allocated dynamically at each iteration of the algorithm, the group size (gs) is a constant set at initialization, Table 1 describes example configurations of group sizes and populations.

The GB mechanic takes a minimalist and dynamic approach to the implementation of the grouping mechanism within the nature-inspired algorithm, maintaining as much of the standard implementation and computational simplicity of the original algorithm as possible, while offering an increase in search diversity. The formation of groups in the group-based augmentation is shown in (15), where G represents the group size, k the group index (0 to $\frac{population}{G}-1$), $x_{k\cdot G}$ is the first agent in the k-th group, $x_{k\cdot G+1}$ is the next agent in the group, and $x_{(k+1)\cdot G-1}$ is the final population member in the group.

$$Group_k = \{x_{k\cdot G}, x_{k\cdot G+1}, \ldots, x_{(k+1)\cdot G-1}\} \tag{15}$$

The group-based augmentation maintains dynamic but distinct groupings of population members at each iteration, which promotes each group functioning independently before groupings are reallocated at the next iteration. This can help prevent premature convergence through the increased search diversity offered. Additionally, through structuring the population into groupings at each iteration, the augmentation can implement a level of balance between exploration and exploitation, which is particularly useful in complex optimization problems which require exploration of different regions simultaneously.

Cross-group framework

The second novel group-based paradigm presented in this research is the XGB approach, which implements a cross collaboration between the groups within the population. As with the GB approach, groupings are dynamically assigned at each iteration, along with group leaders (x_{gl}), with the caveat that the final group member in each group (apart from the last group), is the leader for the next population group. As the groups overlap, it alters the movement of the population and increases search diversity within the problem domain, offering additional flexibility that the group-based augmentation may lack in some problem domains.

The XGB approach once again prioritizes a minimalist and dynamic approach to the implementation of the grouping mechanics, attempting to maintain the overall computational simplicity of the algorithm it has been augmented to, while ultimately increasing search diversity through varying the movement behavior of individual agents within the population. Table 2 describes examples of different configurations of group sizes and populations for the cross group-based approach. The formation of groups within this augmentation is expressed in (16), with G representing the group size, k the group index (0 to $\frac{population}{G}-1$), and $x_{k+(G-1)}$ is the final population member in the group.

$$Group_k = \{x_k, x_{k+1}, \ldots, x_{k+(G-1)}\} \tag{16}$$

Table 1 Group-based augmentation grouping examples

Group size (gs)	Population (n)	Group leaders
3	30	$\{x_1,\ x_4, \ldots, x_{25}, x_{28}\}$
5	50	$\{x_1,\ x_6, \ldots, x_{41}, x_{46}\}$
10	80	$\{x_1,\ x_{11}, \ldots, x_{61}, x_{71}\}$

Table 2 Cross group-based augmentation grouping examples

Group size (gs)	Population (n)	Group leaders
4	40	$\{x_1,\ x_4, \dots, x_{34}, x_{37}\}$
8	80	$\{x_1,\ x_8, \dots, x_{64}, x_{72}\}$
10	100	$\{x_1,\ x_{10}, \dots, x_{80}, x_{90}\}$

The Cross-Group augmentation facilitates additional interaction between different parts of the population, which can increase search diversity and flexibility within the population, to adapt to complex fitness landscapes. Additionally, by being part of multiple groups, population members can explore the search space in a more diverse way, while still exploiting the solutions identified by the group leaders.

GB and XGB algorithm pseudocode and movement equations

When described as pseudocode, the functionality of the GB and XGB algorithms is fundamentally the same, with the primary difference being how group sizes are managed and how group leaders are assigned, and a slightly updated movement equation. This subsection presents the pseudocode and updated movement equation for each of the proposed framework variants.

GBBA and XGBBA

The updated GBBA and XGBBA velocity update equation is shown in (17), with x_l^t representing the position of the group leader, with x_i^t moving toward the group leader if a better fitness is offered.

$$v_i^{t+1} = v_i^t + \left(x_i^t - x_l^t\right) f_i \tag{17}$$

The pseudo code for GBBA and XGBBA can be seen in Algorithm 1.

Algorithm 1: Pseudocode for GBBA and XGBBA

```
1:  Initialize population $x_i$ and $v_i$ ( $i = 1,2, \dots, n$)
2:  Initialize frequencies $f_i$, pulse rates $r_i$ and loudness $A_i$
3:  Assign groupings and group leaders ( $g_n$) based on GB or
    XGB augmentation
4:  while ( $t$ <Max_Iterations)
5:  Generate new solutions by adjusting frequency
6:  Update velocities and positions/solutions
7:  **if** (rand $> r_i$)
8:  Select a solution among the best solutions
9:  Generate a local solution around the selected best solution
10: **end if**
11: Generate a new solution by flying toward group leader
    ( $g_n$)
12: **if** ( $g_n < A_i$ & $f(x) < f(x)$)
13: Accept the new solutions
14: Increase $r_i$ and reduce $A_i$
15: If flying toward group leader is not accepted, generate a
    new solution by flying randomly
16: **else** (rand $< A_i$ & $f(x) < f(x)$)
17: Accept the new solutions
18: Increase $r_i$ and reduce $A_i$
19: **end if**
20: Rank bats and find the current best $x_*$
21: Assign new groupings and group leaders ( $g_n$) based on
    GB or XGB augmentation
22: **end while**
```

GBFA and XGBFA

The updated GBFA and XGBFA movement equation is shown in (18), with x_l^t representing the position of the group leader, with x_i^t moving toward the group leader if it has a higher brightness.

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}\left(x_l^t - x_i^t\right) + \alpha\, \epsilon_i^t \tag{18}$$

The pseudo code for GBFA and XGBFA is shown in Algorithm 2.

Algorithm 2: Pseudocode for GBFA and XGBFA

```
1:  Define objective function : $f(x)$ , $x = (x_1, \dots, x_d)^T$
2:  Initialize population of $n$ fireflies: $x_i (i = 1,2, \dots, n)$
3:  Light intensity $I_i$ at $x_i$ is determined by $f(x_i)$
4:  Define light absorption coefficient $\gamma$
5:  Assign groupings and group leaders ( $g_n$) based on
    GB or XGB augmentation
6:  **while** ( $t$ <MaxGeneration)
7:  **for** $i = 1 : n$ (all $n$ fireflies)
8:  **for** $j = 1 : n$(all $n$ fireflies ) (inner loop)
9:  Move toward group leader if better:
10: **if** ( $I_i < g_n$)
11: Move firefly $i$ towards $g_n$
12: Move as normal if group leader not better:
13: **else** ( $I_i < I_j$)
14: Move firefly $i$ towards $j$
15: **end if**
16: Vary attractiveness with distance $r$ via $exp[-\gamma r^2]$
17: Evaluate new solutions and update light intensity
18: **end for** j
19: **end for** $i$
20: Rank the fireflies and find current global best $g_*$
21: Assign new groupings and group leaders ( $g_n$) based
    on GB or XGB augmentation
22: **end while**
```

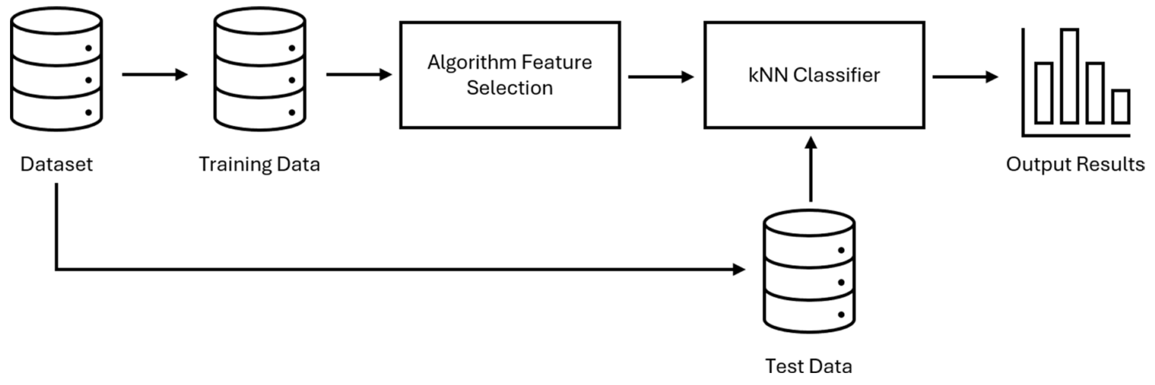

Fig. 1 System overview

GBPSO and XGPSO

The updated GBPSO and XGBPSO movement equation is shown in (19), with x_l^t representing the position of the group leader, with x_i^t moving toward the group leader if it has a better fitness.

$$v_i^{t+1} = v_i^t + \alpha\,\epsilon_1\left[g^* - x_i^t\right] + \beta\,\epsilon_2[x_l^{(t)} - x_i^t] \tag{19}$$

Pseudo code for the GBPSO and the XGBPSO can be seen in Algorithm 3.

Algorithm 3: Pseudocode for GBPSO and XGBPSO

```
1:  Define objective function : $f(x)$, $x = (x_1, \ldots, x_d)^T$
2:  Initialize velocity ($v_i$) and location ($x_i$) of $n$ particles
3:  Find global best ($g^*$) at $t = 0$:
    $g^* = \min\{f(x_1), \ldots, f(x_n)\}$
4:  Assign groupings and group leaders ($g_n$) based on GB
    or XGB augmentation
5:  while (criterion)
6:  for loop over all particles ($n$) and dimensions ($d$)
7:  Generate new velocity ($v_i^{t+1}$) using Eq. (13), using
    group leader ($g_n$) instead of global best ($g^*$)
8:  Calculate new locations ($x_i^{t+1} = x_i^t + v_i^{t+1}$)
9:  Evaluate objective function at new locations ($x_i^{t+1}$)
10: Find the current best for each particle ($x_i^*$)
11: end for
12: Find the current global best ($g^*$)
13: Update iteration or pseudo time ($t = t + 1$)
14: Assign new groupings and group leaders ($g_n$) based
    on GB or XGB augmentation
15: end while
```

Experimentation overview

A system was built using Python 3.10, which implements the base versions of the BA, FA and PSO, along with the six proposed algorithms, to perform feature selection, with a k-NN used for classification, with the functionality of the system visualized in Fig. 1.

The algorithms were tested using 21 datasets from the UCI machine learning repository, which are commonly used to test the performance of feature selection and classification systems [4], described in Table 3. Datasets were chosen based on the number of features and samples, to represent testing in different problem domains (small, medium and large), with each dataset being split into training and testing data. To maintain consistency across all experimentation, and with other current research within the field, each algorithm was configured using the parameters: 20 runs, 1000 iterations, population size of 20, group size of 4.

Table 3 UCI machine learning repository datasets

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	365	1593
18	Clean 1	166	476
19	Clean 2	166	6598
20	Lymphography	18	148
21	PenglungEW	325	73

Results

Each of the base algorithms and proposed algorithms were tested against the 21 UCI machine learning datasets of varying complexity, for 20 runs at 1000 iterations each run. The average results for accuracy, precision, recall and F1 score of each algorithm are presented and discussed in this section, along with a comparison between the GBBA, XGBBA, GBFA, XGBFA, GBPSO and XGBPSO algorithms proposed in this study. Later in this section, the average accuracy performance of the proposed group-based algorithms is compared to other modern feature selection algorithms and methods: bWOA-S and bWOA-v presented in [9], the Interaction-Guided Incremental Selection (IGIS) presented in [17], Interaction Gain – Recursive Feature Elimination (IG-RFE) method proposed in [13] and the k-NN rough set reduction (k-NNRS) method presented in [24].

Average accuracy

Table 4 shows the average accuracy for BA, GBBA, XGBBA, FA, GBFA, XGBFA, PSO, GBPSO and XGBPSO for all 21 UCI machine learning datasets, with the data visualized in Fig. 2. In terms of average accuracy, all proposed algorithms show positive results and outperform their standard implementations, with the GBPSO and XGBPSO algorithms demonstrating the best overall performance.

The BA algorithm performed with an average accuracy of 0.79, with the GBBA averaging 0.82 and presenting the best of the BA variants, and XGBBA showing a 0.81 average accuracy. The FA algorithm demonstrated an average accuracy of 0.78 across all datasets, with GBFA performing the best within the group, presenting an accuracy of 0.81, and XGBFA performed with an average accuracy of 0.80. Finally, the PSO algorithm showed an average accuracy of 0.85, with the GBPSO and XGBPSO demonstrating an 0.88 average accuracy. Interestingly, while all algorithm variants improved upon the performance of their standard implementations, the standard deviation of average accuracy across all datasets remains at a consistent level with the standard implementation for each proposed algorithm. The BA, GBBA and XGBBA all have a standard deviation of 0.09. FA has a standard deviation of 0.10, with the GBFA and XGBFA reducing this slightly to 0.09. PSO, GBPSO and XGBPSO all demonstrated a standard deviation of 0.07.

Average precision

The average precision of each algorithm is shown in Table 5 and is visualized in Fig. 3, with all algorithm variants proposed in this paper performing better than their standard implementations. BA returned an average precision of 0.78

Table 4 Average accuracy

Group	Algorithm	Dataset																					*Avg.*
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
BA	BA	0.95	0.69	0.80	0.73	0.79	0.72	0.85	0.64	0.87	0.78	0.75	0.65	0.70	0.73	0.89	0.88	0.88	0.83	0.95	0.71	0.84	*0.79*
	GBBA	0.97	0.72	0.82	0.84	0.80	0.73	0.87	0.66	0.91	0.79	0.76	0.68	0.72	0.76	0.90	0.91	0.87	0.84	0.96	0.73	0.89	*0.82*
	XGBBA	0.97	0.71	0.85	0.75	0.76	0.79	0.85	0.65	0.93	0.80	0.76	0.65	0.71	0.77	0.89	0.93	0.89	0.85	0.96	0.78	0.85	*0.81*
FA	FA	0.94	0.65	0.80	0.69	0.75	0.75	0.85	0.58	0.86	0.73	0.74	0.64	0.70	0.71	0.86	0.91	0.88	0.83	0.96	0.69	0.83	*0.78*
	GBFA	0.95	0.68	0.83	0.80	0.83	0.81	0.87	0.60	0.89	0.78	0.75	0.67	0.72	0.75	0.91	0.92	0.89	0.87	0.96	0.76	0.88	*0.81*
	XGBFA	0.95	0.68	0.83	0.75	0.81	0.78	0.86	0.63	0.88	0.80	0.74	0.64	0.72	0.74	0.86	0.92	0.90	0.85	0.96	0.76	0.84	*0.80*
PSO	PSO	0.96	0.78	0.85	0.86	0.82	0.84	0.85	0.74	0.95	0.87	0.81	0.72	0.75	0.84	0.91	0.92	0.92	0.90	0.97	0.82	0.86	*0.85*
	GBPSO	0.98	0.79	0.90	0.91	0.82	0.89	0.89	0.74	0.97	0.89	0.81	0.74	0.75	0.85	0.95	0.92	0.92	0.91	0.97	0.87	0.91	*0.88*
	XGBPSO	0.97	0.79	0.92	0.85	0.85	0.85	0.89	0.77	0.95	0.91	0.81	0.74	0.75	0.85	0.97	0.95	0.91	0.91	0.97	0.85	0.94	*0.88*

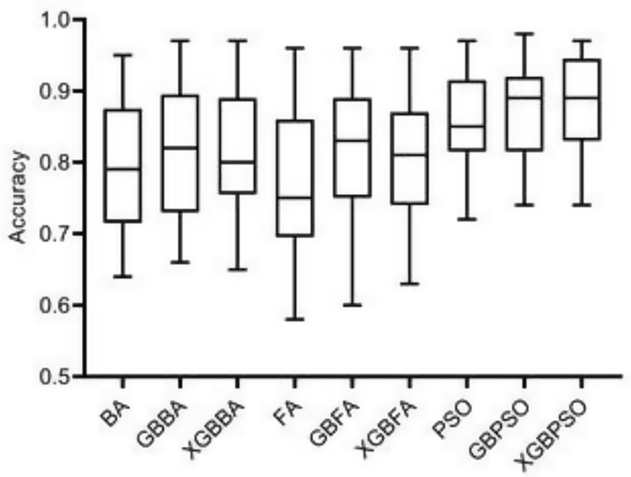

Fig. 2 Average accuracy of BA, GBBA, XGBBA, FA, GBFA, XGBFA, PSO, GBPSO and XGBPSO

across all datasets, GBBA demonstrated the best performance of the algorithm variants, with an average of 0.81 and XGBBA showed an average of 0.80. Both the GBFA and XGBFA demonstrated an average performance of 0.80, with the standard FA returning 0.76. The GBPSO and XGBPSO both demonstrated an average precision of 0.88, with the standard PSO showing an average precision of 0.86.

Average recall

Table 6 presents the average recall performance of all algorithms, with the data visualized in Fig. 4. GBBA and XGBBA demonstrated the same average recall performance with 0.76, and performed better than the standard BA implementation, which showed a performance of 0.74. The GBFA and XGBFA both returned an average performance of 0.75, with FA presenting an average of 0.71. GBPSO presented the best average recall of the PSO group, with the XGBPSO performing worse of the group, with an average recall of 0.75 recorded, which is also reflected in Fig. 4, where XGBPSO shows a lower center of distribution than the standard PSO implementation, which achieved an average recall of 0.80.

Average F1 score

Table 7 shows the average F1 score, with all algorithm variants proposed in this paper outperforming their respective standard implementations, with Fig. 5 visualizing the data. GBBA and XGBBA both showed an average F1 score of 0.76, with the standard BA implementation returning an average F1 score of 0.74. The standard FA presents an average F1 score of 0.72, with GBFA and XGBFA improving upon this with an average of 0.75. Both the GBPSO and

Table 5 Average precision

Group	Algorithm	Dataset																					*Avg.*
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
BA	BA	0.94	0.65	0.82	0.76	0.69	0.73	0.89	0.63	0.87	0.78	0.75	0.55	0.54	0.72	0.89	0.89	0.89	0.84	0.93	0.84	0.87	*0.78*
	GBBA	0.97	0.68	0.83	0.85	0.68	0.74	0.90	0.66	0.91	0.79	0.76	0.60	0.56	0.74	0.90	0.91	0.88	0.84	0.94	0.87	0.94	*0.81*
	XGBBA	0.96	0.68	0.85	0.76	0.67	0.80	0.89	0.65	0.93	0.80	0.76	0.58	0.55	0.76	0.89	0.94	0.89	0.85	0.93	0.89	0.87	*0.80*
FA	FA	0.94	0.61	0.81	0.68	0.57	0.77	0.89	0.57	0.86	0.73	0.74	0.54	0.53	0.69	0.85	0.91	0.89	0.84	0.94	0.85	0.80	*0.76*
	GBFA	0.94	0.64	0.83	0.81	0.76	0.82	0.89	0.60	0.88	0.78	0.75	0.57	0.54	0.73	0.90	0.92	0.89	0.86	0.94	0.88	0.86	*0.80*
	XGBFA	0.94	0.65	0.84	0.76	0.72	0.79	0.91	0.63	0.87	0.80	0.74	0.54	0.57	0.72	0.86	0.92	0.91	0.86	0.94	0.89	0.87	*0.80*
PSO	PSO	0.95	0.75	0.87	0.87	0.78	0.87	0.86	0.74	0.94	0.87	0.81	0.72	0.74	0.83	0.90	0.92	0.93	0.90	0.96	0.91	0.92	*0.86*
	GBPSO	0.97	0.78	0.90	0.92	0.74	0.90	0.92	0.74	0.96	0.90	0.81	0.70	0.81	0.85	0.95	0.93	0.93	0.91	0.96	0.94	0.93	*0.88*
	XGBPSO	0.97	0.78	0.91	0.86	0.81	0.85	0.91	0.77	0.95	0.91	0.81	0.73	0.76	0.84	0.96	0.94	0.92	0.91	0.96	0.93	0.96	*0.88*

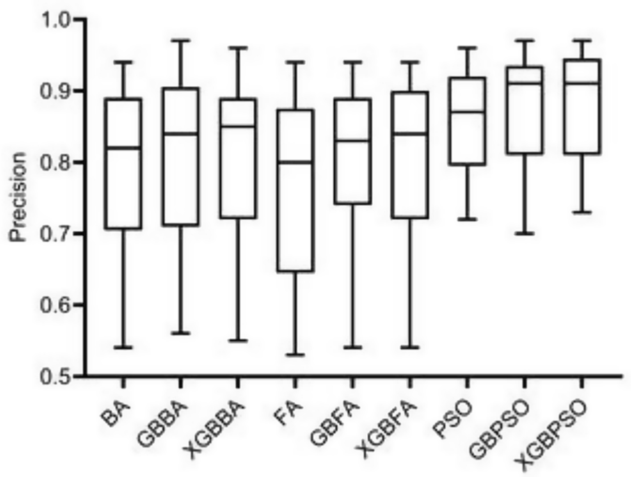

Fig. 3 Average precision of BA, GBBA, XGBBA, FA, GBFA, XGBFA, PSO, GBPSO and XGBPSO

XGBPSO obtained an average F1 score of 0.82 and the standard PSO implementation achieved an average F1 score of 0.79. It also should be noted that while the XGBPSO demonstrated a higher overall average F1 score than the standard PSO implementation, the center of distribution shown in Fig. 5 is lower than the standard PSO implementation.

Optimization time

Table 8 shows the average optimization time of the standard algorithm implementations and the algorithm variants from the proposed framework, with the data visualized in Fig. 6. As can be expected through modification of the movement and attraction behaviors of the algorithms, the complexity of the algorithm increases in most cases, with both the GB approach and XGB approach increasing the optimization time in the GBBA, XGBBA, XGBFA, GBPSO and XGBPSO implementations. While the complexity of these algorithm variants is increased from the base implementation, they are also shown to offer a better overall performance.

Due to the computational simplicity of the algorithm, the PSO implementation provided the best result on 17 of the datasets, with the GBPSO and variant showing an average increase of 4.17 s in optimization time, and the XGBPSO showing an average increase of 12.96 s. Therefore, we can conclude that while XGBPSO offers the best performance of the three PSO algorithms, it also comes with an increased computational cost. Interestingly, in the GBFA implementation, the complexity of the algorithm has been reduced in comparison to the base FA implementation across, with the base FA implementation achieving an average optimization time of 21.12 s, and GBFA showing a 4.19 s decrease with an average of 16.93 s. This can be attributed to the lower complexity of the movement rules of the GBFA implementation

Table 6 Average recall

Group	Algorithm	Dataset																					*Avg.*
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
BA	BA	0.94	0.64	0.68	0.73	0.71	0.72	0.80	0.63	0.88	0.78	0.75	0.54	0.53	0.71	0.90	0.86	0.88	0.84	0.89	0.37	0.86	*0.74*
	GBBA	0.97	0.66	0.64	0.84	0.64	0.72	0.82	0.66	0.91	0.79	0.76	0.59	0.52	0.73	0.90	0.91	0.87	0.85	0.90	0.38	0.84	*0.76*
	XGBBA	0.96	0.67	0.73	0.74	0.66	0.79	0.80	0.65	0.93	0.80	0.76	0.57	0.51	0.75	0.89	0.92	0.89	0.85	0.90	0.41	0.84	*0.76*
FA	FA	0.94	0.60	0.65	0.68	0.53	0.74	0.80	0.57	0.87	0.73	0.74	0.53	0.52	0.68	0.86	0.90	0.88	0.84	0.90	0.36	0.71	*0.71*
	GBFA	0.94	0.63	0.65	0.80	0.73	0.80	0.83	0.60	0.89	0.77	0.75	0.55	0.51	0.72	0.90	0.91	0.89	0.87	0.91	0.40	0.79	*0.75*
	XGBFA	0.95	0.64	0.68	0.75	0.70	0.77	0.80	0.62	0.86	0.80	0.74	0.53	0.52	0.71	0.86	0.90	0.90	0.86	0.90	0.39	0.85	*0.75*
PSO	PSO	0.95	0.73	0.72	0.86	0.65	0.83	0.81	0.73	0.95	0.87	0.81	0.60	0.53	0.84	0.92	0.91	0.92	0.91	0.92	0.51	0.82	*0.80*
	GBPSO	0.98	0.73	0.82	0.91	0.73	0.89	0.85	0.73	0.97	0.89	0.81	0.65	0.50	0.83	0.96	0.90	0.92	0.91	0.92	0.48	0.91	*0.82*
	XGBPSO	0.95	0.64	0.68	0.75	0.70	0.77	0.80	0.62	0.86	0.80	0.74	0.53	0.52	0.71	0.86	0.90	0.90	0.86	0.90	0.39	0.85	*0.75*

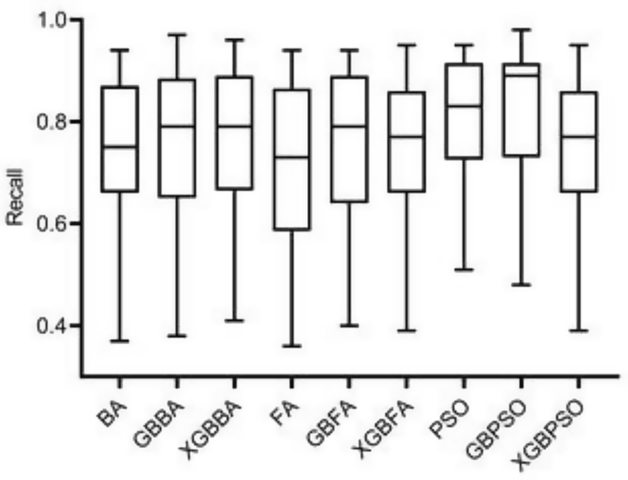

Fig. 4 Average recall of BA, GBBA, XGBBA, FA, GBFA, XGBFA, PSO, GBPSO and XGBPSO

when compared to the standard FA implementation, with the GBFA impacting the attraction behaviors of the algorithm. The XGBFA, however, does show an increase in computational complexity, with an overall average increase of 11.55 s. This increase can be seen amongst all XGB algorithm variants and can be attributed to the increased amount of groups within the population, and the associated computational cost caused by this.

Feature selection performance and algorithm stability

This section provides a brief review of feature selection performance and algorithm stability of the proposed algorithm variants, with Table 9 showing the average amount of selected features, Table 10 showing the average minimum amount of selected features, and Table 11 showing the average maximum selected features. The average amount of selected features is consistently higher in all GB and XGB augmented algorithms, and given that the performance of these implementations generally improves upon the quality of the base implementations. While the number of features is not a clear indication of algorithm robustness or quality, it does provide provides an indication that quality features were selected and therefore the dimensionality has potentially been reduced by ignoring irrelevant or redundant features.

Jaccard index

The stability of the proposed algorithms has been evaluated using the Jaccard Index, with the similarity coefficient for each algorithm and dataset shown in Table 12. The Jaccard Index has been used to evaluate the robustness and stability

Table 7 F1 score

Group	Algorithm	Dataset																					*Avg.*
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
BA	BA	0.94	0.64	0.66	0.73	0.69	0.72	0.82	0.63	0.87	0.78	0.75	0.54	0.52	0.71	0.89	0.87	0.88	0.83	0.91	0.37	0.85	*0.74*
	GBBA	0.97	0.66	0.62	0.84	0.70	0.72	0.85	0.66	0.91	0.79	0.76	0.59	0.50	0.73	0.90	0.91	0.87	0.84	0.92	0.37	0.85	*0.76*
	XGBBA	0.96	0.67	0.71	0.74	0.65	0.79	0.82	0.65	0.93	0.80	0.76	0.57	0.50	0.75	0.89	0.93	0.88	0.84	0.92	0.40	0.81	*0.76*
FA	FA	0.94	0.60	0.62	0.67	0.60	0.74	0.82	0.57	0.86	0.73	0.74	0.53	0.51	0.68	0.85	0.90	0.88	0.83	0.92	0.35	0.83	*0.72*
	GBFA	0.94	0.63	0.64	0.80	0.72	0.80	0.85	0.59	0.89	0.77	0.75	0.55	0.48	0.72	0.90	0.91	0.89	0.86	0.93	0.39	0.84	*0.75*
	XGBFA	0.95	0.64	0.64	0.74	0.69	0.78	0.83	0.62	0.87	0.80	0.74	0.51	0.51	0.72	0.86	0.91	0.90	0.85	0.92	0.38	0.86	*0.75*
PSO	PSO	0.95	0.73	0.67	0.86	0.66	0.84	0.83	0.73	0.94	0.87	0.81	0.57	0.49	0.83	0.91	0.92	0.92	0.90	0.94	0.50	0.82	*0.79*
	GBPSO	0.97	0.77	0.79	0.91	0.73	0.89	0.87	0.73	0.97	0.89	0.81	0.63	0.44	0.83	0.95	0.91	0.92	0.91	0.94	0.47	0.91	*0.82*
	XGBPSO	0.97	0.75	0.81	0.86	0.72	0.85	0.87	0.76	0.95	0.91	0.81	0.61	0.47	0.84	0.96	0.94	0.91	0.91	0.93	0.51	0.92	*0.82*

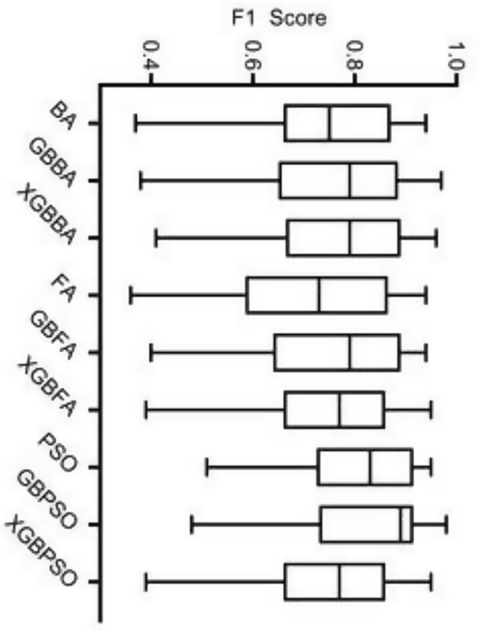

Fig. 5 Average F1 Score of BA, GBBA, XGBBA, FA, GBFA, XGBFA, PSO, GBPSO and XGBPSO

of the proposed algorithm variants, and is a statistical test used to measure the similarity and diversity of sample sets, in this case, selected features. The Jaccard Index can be expressed as (20), with J representing the Jaccard Index, A as set 1 and B as set 2.

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|} \tag{20}$$

Overall, the stability of the proposed algorithm variants that incorporate the GB and XGB frameworks have shown an improved algorithm stability when used for feature selection, with eighteen of the algorithm variants showing an increased stability. The algorithm variants XGBBA, XGBFA and XGBPSO are noted as the most stable performers, with each showing the highest performance on four different datasets. The XGBBA algorithm shows the highest overall stability, with a Jaccard Index score of 0.392.

Group-based and cross group-based average accuracy comparison

This section will briefly review the performance of the group-based and cross group-based algorithms, using accuracy as the primary comparison metric, as this is the metric that will be used to evaluate the performance of these algorithms against other modern feature selection algorithms. In the next subsection. Figure 7 shows the average accuracy of GBBA, XGBBA, GBFA, XGBFA, GBPSO and XGBPSO on all datasets.

The best performing group-based algorithm was the GBPSO, with an average of 0.88, and a difference of 0.06 higher than the next best performing group-based algorithm, the GBBA, with a 0.82 average. Performing marginally

Table 8 Optimization time

Group	Algorithm	Dataset																					*Avg.*
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
BA	BA	17.44	17.07	16.58	16.73	16.43	16.54	16.48	16.60	16.68	17.75	18.74	17.01	17.52	17.16	16.43	16.70	19.42	17.02	27.04	16.54	16.68	*17.55*
	GBBA	16.68	19.49	15.98	16.38	16.17	16.33	16.39	16.30	16.62	19.33	21.64	18.42	18.75	18.12	16.49	16.63	24.20	17.37	26.46	16.29	16.73	*18.13*
	XGBBA	31.71	33.52	29.90	29.77	29.21	29.98	28.57	29.20	29.79	30.48	25.56	29.27	29.96	32.51	28.98	32.79	32.28	34.05	32.48	28.71	31.15	*30.47*
FA	FA	20.34	20.39	19.37	19.38	19.47	19.68	19.87	19.59	21.02	22.55	25.79	18.77	21.41	20.89	20.80	20.03	28.14	20.90	25.36	19.58	20.22	*21.12*
	GBFA	15.57	18.08	15.23	18.22	12.55	15.42	17.23	15.48	14.55	18.37	17.26	13.71	19.74	19.76	14.80	17.58	18.85	14.92	25.22	15.42	17.61	*16.93*
	XGBFA	35.32	34.83	31.80	30.12	31.65	32.73	32.73	32.21	33.27	29.41	35.56	28.13	29.30	30.82	31.66	34.08	34.48	36.36	35.53	32.37	33.61	*32.67*
PSO	PSO	14.03	14.04	13.14	12.97	13.12	13.14	13.02	13.18	13.20	15.25	17.59	13.73	13.74	13.76	13.15	13.46	19.13	14.03	25.02	12.95	13.46	*14.53*
	GBPSO	17.76	19.00	17.35	18.21	18.06	17.30	17.74	16.81	17.71	19.44	22.48	17.45	17.20	16.20	16.97	17.60	25.03	17.17	27.80	18.37	16.96	*18.70*
	XGBPSO	22.41	25.94	26.67	25.60	25.50	27.27	25.93	27.17	18.40	31.17	33.48	25.48	27.74	31.07	25.48	29.28	35.14	31.94	26.57	25.36	29.62	*27.49*

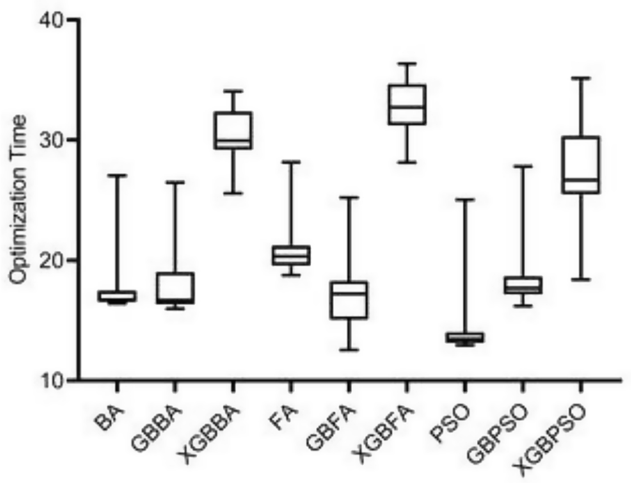

Fig. 6 Average optimization time of BA, GBBA, XGBBA, FA, GBFA, XGBFA, PSO, GBPSO and XGBPSO

worse than the GBBA was the GBFA, with an overall average accuracy of 0.81. In terms of the cross group-based algorithms, the PSO algorithm variant was again the best, with XGBPSO showing an average accuracy of 0.88. The BA based algorithm variant, XGBBA was again the second-best performing algorithm, with an average accuracy of 0.81, and XGBFA showing a marginally lower average accuracy of 0.80. The XGBPSO showed a 0.07 higher average accuracy the XGBBA, and a 0.08 higher average accuracy than the XGBFA. It can also be seen in Fig. 7 that the GBPSO and XGBPSO were the best performing of the proposed algorithms, with a higher center of distribution than GBBA, XGBBA, GBFA and XGBFA. Interestingly, the GBPSO demonstrated a negatively skewed distribution of results, with the center of distribution sitting in the upper quartile.

Group-based and cross group-based average accuracy on low feature data sets

Figure 8 shows a comparison of average accuracy performance of the proposed group-based and cross group-based algorithms on datasets with a feature count below 100 (datasets 1–16 and 20). The GBBA and XGBBA both demonstrated an average accuracy of 0.80, with a very similar center of distribution shown. GBFA and XGBFA were both marginally outperformed by the BA based algorithms, presenting an average accuracy of 0.79, this time showing a slight variance in the center of distribution, as can be seen in Fig. 8. Of the BA and FA based algorithm variants, the center of distribution was quite close, with very slight variances shown. The PSO based algorithm variants were once again the best performers of the group-based and cross group-based algorithms, with both the GBPSO and XGBPSO showing an average accuracy of 0.86, and again showing a

Table 9 Average selected features

Group	Algorithm	Dataset																					*Avg.*
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
BA	BA	2.8	7.1	8.7	5.8	10.8	28.2	12.4	6.0	5.8	17.3	18.9	8.4	6.9	8.3	5.5	11.4	125.7	80.9	72.6	9.7	157.9	*29.1*
	GBBA	3.0	5.6	7.9	5.9	10.5	28.2	13.7	6.5	5.9	18.1	18.7	8.0	7.5	7.4	6.0	11.1	124.4	80.0	73.5	9.3	157.4	*29.0*
	XGBBA	3.0	5.0	8.2	6.2	10.2	28.8	14.3	6.7	6.7	17.6	19.7	7.4	6.0	7.0	6.4	13.3	124.9	79.1	75.6	9.2	158.9	*29.2*
FA	FA	2.3	7.1	7.5	4.7	8.8	24.6	12.8	5.5	4.2	17.1	16.9	8.6	4.9	6.8	4.2	8.9	113.9	66.6	57.3	7.9	123.2	*24.5*
	GBFA	3.0	7.3	8.8	5.8	10.7	27.7	4.4	6.5	5.9	18.6	18.4	8.3	6.6	8.2	5.4	11.1	125.0	79.2	71.7	10.3	155.3	*28.5*
	XGBFA	2.8	7.0	8.5	5.9	10.5	28.3	12.8	6.5	5.6	19.5	19.2	8.8	7.4	8.9	5.7	10.8	125.0	81.3	72.4	9.1	158.0	*29.2*
PSO	PSO	3.6	5.1	7.2	6.2	10.1	28.3	14.6	6.5	6.9	16.9	19.0	6.5	6.2	6.7	7.3	13.4	129.1	80.3	76.6	8.3	154.6	*29.2*
	GBPSO	3.9	6.0	7.4	5.8	11.2	29.4	15.9	6.4	6.6	17.6	18.7	7.0	7.6	7.0	7.1	13.9	131.1	83.3	82.3	9.2	165.9	*30.6*
	XGBPSO	4.4	5.7	8.3	6.9	10.9	28.8	15.5	6.7	8.1	17.8	19.9	6.3	6.8	8.1	8.2	14.4	129.5	84.0	81.1	8.6	164.5	*30.7*

Table 10 Minimum selected features

Group	Algorithm	Dataset																					*Avg.*
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
BA	BA	2	5	5	3	8	22	7	4	3	12	14	7	4	4	3	8	108	67	64	6	142	*23.7*
	GBBA	2	2	3	3	7	22	10	4	3	14	15	4	1	4	2	8	111	67	60	5	139	*23.1*
	XGBBA	2	3	5	4	8	20	7	4	4	14	15	5	2	4	3	9	111	67	63	5	143	*23.7*
FA	FA	2	6	4	2	5	17	8	3	1	10	10	7	1	4	1	5	73	46	45	3	102	*16.9*
	GBFA	2	6	7	3	7	20	2	4	3	14	13	7	1	4	3	6	113	64	56	7	142	*23.0*
	XGBFA	2	7	6	3	6	24	8	3	2	14	14	7	1	6	4	6	118	68	63	7	136	*24.0*
PSO	PSO	1	3	3	3	6	20	6	1	4	12	11	3	2	3	1	7	109	61	58	2	134	*21.4*
	GBPSO	2	4	3	3	8	21	5	4	1	11	14	4	3	3	2	6	111	63	65	6	147	*23.1*
	XGBPSO	2	2	5	4	8	22	6	4	3	11	14	5	3	6	4	7	108	72	65	5	137	*23.5*

Table 11 Maximum selected features

Group	Algorithm	Dataset																					*Avg.*
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
BA	BA	4	9	12	8	14	33	18	9	8	24	25	12	10	11	8	15	141	92	80	15	178	*34.6*
	GBBA	7	9	10	10	15	35	20	11	8	22	22	10	10	11	13	19	143	92	96	14	184	*36.2*
	XGBBA	7	8	12	9	14	35	20	11	11	23	27	11	10	11	12	20	136	89	94	14	183	*36.0*
FA	FA	4	9	12	7	16	33	18	9	7	22	24	12	8	10	5	9	140	81	72	12	167	*32.2*
	GBFA	4	9	11	8	14	36	8	10	8	25	24	10	10	11	8	14	139	95	80	15	173	*33.9*
	XGBFA	4	8	11	8	15	35	18	10	9	22	25	12	10	11	8	15	144	94	81	13	176	*34.7*
PSO	PSO	7	9	11	9	15	36	21	11	11	25	28	10	10	11	11	17	144	95	89	12	177	*36.1*
	GBPSO	8	9	12	10	15	39	24	10	12	24	26	9	10	9	11	22	154	97	96	13	180	*37.6*
	XGBPSO	8	9	14	10	16	37	20	11	12	25	26	10	9	11	13	19	149	94	96	13	178	*37.1*

Table 12 Jaccard index

Group	Algorithm	Dataset																					*Avg.*
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
BA	BA	0.256	0.688	0.392	0.377	0.352	0.311	0.234	0.311	0.297	0.351	0.346	0.550	0.364	0.551	0.278	0.248	0.328	0.328	0.280	0.402	0.320	*0.360*
	GBBA	0.318	0.442	0.349	0.401	0.353	0.317	0.261	0.344	0.303	0.394	0.462	0.502	0.404	0.521	0.289	0.241	0.323	0.321	0.284	0.383	0.322	*0.359*
	XGBBA	0.273	0.689	0.432	0.348	0.358	0.322	0.267	0.335	0.304	0.338	0.347	0.546	0.455	0.607	0.277	0.285	0.534	0.518	0.307	0.376	0.323	*0.392*
FA	FA	0.390	0.698	0.337	0.302	0.258	0.137	0.245	0.345	0.672	0.323	0.320	0.580	0.096	0.434	0.255	0.130	0.283	0.125	0.055	0.329	0.077	*0.304*
	GBFA	0.338	0.714	0.436	0.346	0.349	0.310	0.106	0.356	0.313	0.396	0.349	0.547	0.325	0.564	0.280	0.232	0.325	0.313	0.290	0.427	0.378	*0.366*
	XGBFA	0.339	0.789	0.419	0.364	0.364	0.373	0.257	0.366	0.288	0.417	0.352	0.559	0.413	0.589	0.306	0.220	0.326	0.371	0.279	0.526	0.320	*0.392*
PSO	PSO	0.325	0.458	0.291	0.323	0.314	0.312	0.268	0.295	0.280	0.397	0.340	0.374	0.362	0.404	0.297	0.285	0.337	0.319	0.301	0.306	0.312	*0.329*
	GBPSO	0.336	0.725	0.327	0.326	0.352	0.327	0.301	0.449	0.257	0.393	0.314	0.423	0.432	0.471	0.339	0.291	0.344	0.339	0.329	0.484	0.342	*0.376*
	XGBPSO	0.294	0.545	0.371	0.393	0.370	0.354	0.322	0.364	0.335	0.360	0.331	0.397	0.342	0.575	0.345	0.305	0.338	0.337	0.322	0.362	0.339	*0.367*

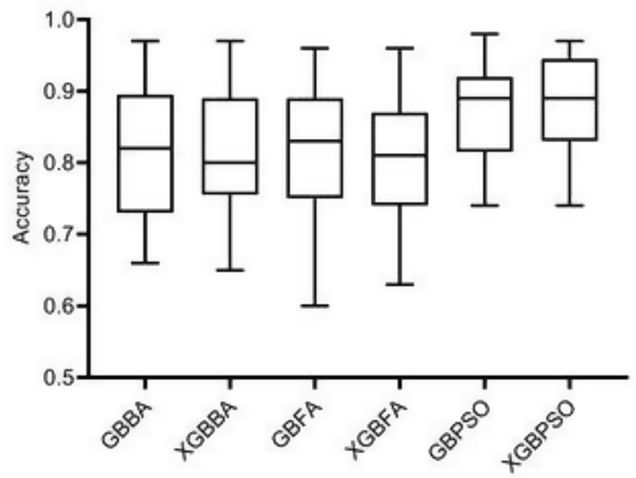

Fig. 7 Average accuracy of GBBA, XGBBA, GBFA, XGBFA, GBPSO and XGBPSO

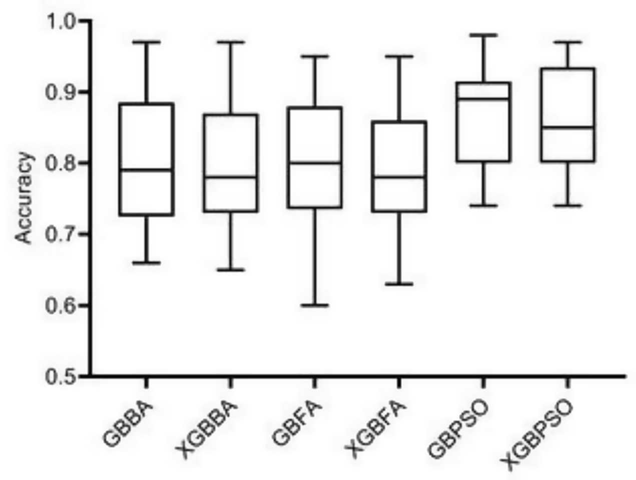

Fig. 8 Average accuracy of GBBA, XGBBA, GBFA, XGBFA, GBPSO and XGBPSO on datasets 1–16 and 20, with less than 100 features

similar center of distribution. The GBPSO showed a negatively skewed distribution, with the center of distribution being close to the upper quartile, while XGBPSO showed a somewhat positively skewed distribution.

Group-based and cross group-based average accuracy on high feature data sets

A comparison of average accuracy performance for the proposed group-based and cross group-based algorithms on datasets with a feature count of greater than 100 (datasets 17–19 and 21) can be seen in Fig. 9.

GBBA demonstrated the best performance of the BA variants with an average accuracy of 0.82, performing marginally better than XGBBA, which presented an average accuracy of 0.81. Of the FA variants, the group-based

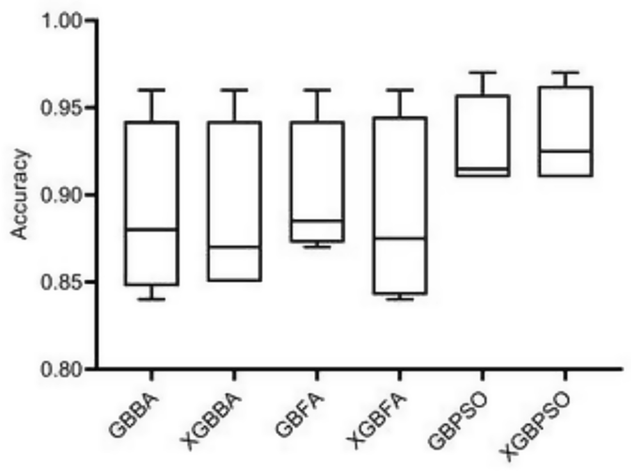

Fig. 9 Average accuracy of GBBA, XGBBA, GBFA, XGBFA, GBPSO and XGBPSO on datasets 17–19 and 21, with more than 100 features

algorithm was again the better performer of the two, with an average accuracy of 0.81, with the XGBFA obtaining an average accuracy of 0.80. Once again, both the PSO algorithm variants demonstrated the best overall performance of the group-based and cross group-based algorithms, with both GBPSO and XGBPSO presenting an average accuracy of 0.88. Again, a pattern can be seen in Fig. 9 in the center of distribution for the BA and FA based algorithm variants, with a marginal difference between them. Interestingly, the GBPSO demonstrated a positively skewed distribution on the datasets with a larger feature count, while the XGBPSO showed a positively skewed distribution, as it did with the low feature count datasets.

Comparison of group-based and cross group-based average accuracy with modern feature selection algorithms

This section presents a comparison of the average accuracy results achieved by the proposed feature selection algorithm variants. Five modern feature selection algorithms were selected for comparison to evaluate the overall robustness and performance of the GBBA, XGBBA, GBFA, XGBFA, GBPSO and XGBPSO. The first two algorithms chosen for comparison are based upon the whale optimization algorithm (WOA), the bWOA-S and bWOA-v. Both are binary variants of the WOA presented in [4], with bWOA-S implementing a Sigmoid transfer function to convert the WOA values to binary, and the bWOA-v using a hyperbolic tangent transfer function. The third algorithm chosen for comparison is the Interaction-Guided Incremental Selection (IGIS) algorithm, proposed in [17], which combines a filter approach to find candidate features, and a wrapper approach by only selecting a feature if adding it to the currently selected subset improves the accuracy significantly. The fourth algorithm selected for comparison is the Interaction Gain – Recursive Feature Elimination (IG-RFE) method [13], which evaluates the importance of features by combing the relevance between the feature and class label, and the interaction among the features. The final feature selection algorithm chosen for comparison is the k-NN rough set reduction (k-NNRS) method proposed in [24]. The k-NNRS implements a model that combines both the advantages of the δ-neighbourhood and the k-NN for feature selection. Table 8 presents the average accuracy of bWOA-S and bWOA-V on all datasets used to benchmark the performance of the proposed GBBA, XGBBA, GBFA, XGBFA, GBPSO and XGBPSO. Table 8 also shows the average accuracy of the IGIS, IG-RFE and k-NNRS for datasets 2–3, 6–8, 16 and 20.

Average accuracy comparison with bWOA-S and bWOA-v

As can be seen from Table 13, and Fig. 10, the proposed algorithm variants (GBBA, XGBBA, GBFA, XGBFA, GBPSO and XGBPSO), all outperformed the bWOA-S (0.74) and bWOA-v (0.73). XGBFA demonstrated the lowest performance of all proposed algorithms, with an average of 0.80, but outperformed the bWOA-S by 0.06 and bWOA-v by 0.07. XGBBA and GBFA both obtained an average of 0.81, outperforming bWOA-S by 0.07 and the bWOA-v by 0.08. GBBA obtained an overall average of 0.82, outperforming the bWOA-S by 0.08 and bWOA-v by 0.09. Finally, both PSO algorithm variants, the GBPSO and XGBPSO achieved an overall average accuracy of 0.88, with both outperforming the bWOA-S by 0.14 and bWOA-v by 0.15. As can be seen in Fig. 10, both the bWOA-S and bWOA-v showed a large variance between the minimum and maximum values

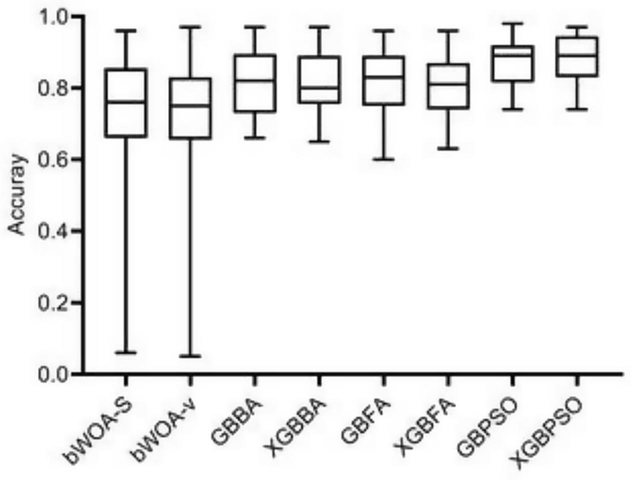

Fig. 10 Average accuracy of bWOA-S, bWOA-v, GBBA, XGBBA, GBFA, XGBFA, GBPSO and XGBPSO on all datasets

achieved, but obtained a relatively central center of distribution. Both the bWOA-S and bWOA-v also demonstrated a relatively large standard deviation of 0.19, while the GBBA, XGBBA, GBFA and XGBFA demonstrated a standard deviation of 0.09, and the GBPSO and XGBPSO both showed a standard deviation of 0.07, demonstrating the robustness of the proposed algorithms on varying sizes and types of datasets (see Table 13).

Average accuracy comparison with IGIS, IG-RFE and k-NNRS

Table 13 shows the average accuracies of the IGIS, IG-RFE and k-NNRS feature selection methods on datasets 2–3, 6–8, 16 and 20, with this data visualized in Fig. 11. In terms of average accuracy across the datasets, IGIS and IG-RFE were the best performing of the comparison group, with both achieving an average accuracy of 0.87, this is 0.01 higher than the average accuracy of the best performing of the proposed algorithm variants, the GBPSO and XGBPSO, which both achieved an average accuracy of 0.86. While the GBPSO and XGBPSO were marginally outperformed by the IGIS and IG-RFE feature selection methods, it should be noted that both proposed algorithm variants demonstrated a slightly lower standard deviation of 0.06, while IGIS showed a standard deviation of 0.07 and IG-RFE demonstrated a standard deviation of 0.8. The k-NNRS algorithm achieved an overall average accuracy of 0.86, with a standard deviation of 0.08, and the best performing of the proposed algorithm variants (GBPSO and XGBPSO) also achieved an average accuracy of 0.86, with a standard deviation of 0.06.

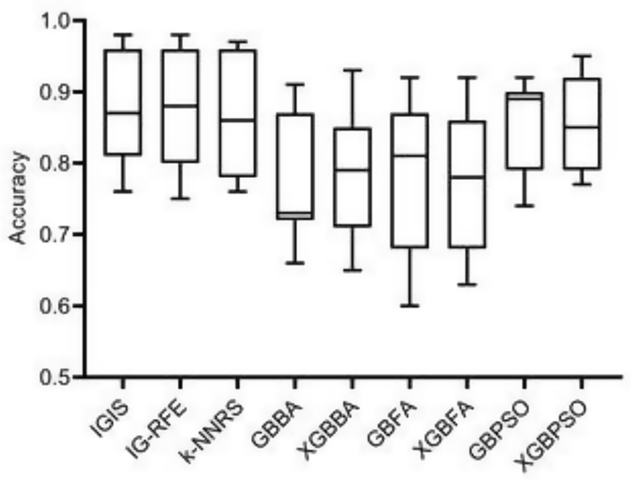

Fig. 11 Average accuracy of bWOA-S, bWOA-v, GBBA, XGBBA, GBFA, XGBFA, GBPSO and XGBPSO datasets 2–3, 6–8, 16 and 20

Table 13 Average accuracy comparison

Group	Algorithm	Dataset																					*Avg.*
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
Other	bWOA-S	0.62	0.80	0.84	0.06	0.67	0.70	0.84	0.65	0.58	0.93	0.81	0.68	0.74	0.87	0.91	0.61	0.96	0.82	0.96	0.76	0.76	*0.74*
	bWOA-v	0.63	0.79	0.82	0.05	0.66	0.70	0.83	0.65	0.60	0.92	0.80	0.69	0.74	0.83	0.90	0.62	0.97	0.80	0.96	0.75	0.73	*0.73*
	IGIS	-	0.83	0.98	-	-	0.88	0.87	0.76	-	-	-	-	-	-	-	0.96	-	-	-	0.81	-	*0.87*
	IG-RFE	-	0.83	0.98	-	-	0.88	0.88	0.75	-	-	-	-	-	-	-	0.96	-	-	-	0.80	-	*0.87*
	k-NNRS	-	0.78	0.97	-	-	0.86	0.89	0.76	-	-	-	-	-	-	-	0.96	-	-	-	0.81	-	*0.86*
BA	GBBA	0.97	0.72	0.82	0.84	0.80	0.73	0.87	0.66	0.91	0.79	0.76	0.68	0.72	0.76	0.90	0.91	0.87	0.84	0.96	0.73	0.89	*0.82*
	XGBBA	0.97	0.71	0.85	0.75	0.76	0.79	0.85	0.65	0.93	0.80	0.76	0.65	0.71	0.77	0.89	0.93	0.89	0.85	0.96	0.78	0.85	*0.81*
FA	GBFA	0.95	0.68	0.83	0.80	0.83	0.81	0.87	0.60	0.89	0.78	0.75	0.67	0.72	0.75	0.91	0.92	0.89	0.87	0.96	0.76	0.88	*0.81*
	XGBFA	0.95	0.68	0.83	0.75	0.81	0.78	0.86	0.63	0.88	0.80	0.74	0.64	0.72	0.74	0.86	0.92	0.90	0.85	0.96	0.76	0.84	*0.80*
PSO	GBPSO	0.98	0.79	0.90	0.91	0.82	0.89	0.89	0.74	0.97	0.89	0.81	0.74	0.75	0.85	0.95	0.92	0.92	0.91	0.97	0.87	0.91	*0.88*
	XGBPSO	0.97	0.79	0.92	0.85	0.85	0.85	0.89	0.77	0.95	0.91	0.81	0.74	0.75	0.85	0.97	0.95	0.91	0.91	0.97	0.85	0.94	*0.88*

Conclusion

This paper has proposed a novel framework, consisting of two approaches to implement group-based mechanisms into nature-inspired optimization algorithms, both of which were designed to increase search diversity to address issues common to nature-inspired metaheuristic algorithms such as premature convergence and oscillations within the population swarms. The results of experimentation demonstrated the robustness of the proposed algorithms, with each proposed variant outperforming the standard implementation across varied datasets. Additionally, all algorithms implemented with the proposed GB and XGB framework showed promise when compared to other modern feature selection algorithms, with all proposed algorithm variants demonstrating an improvement upon the comparison group, or a relatively homogeneous level of performance. While the proposed algorithms have generated promising results, further experimentation is required to fully test the performance, flexibility and robustness of them. Due to the nature of the GB and XGB frameworks and their modification of movement behaviors within nature-inspired optimization algorithms, it is expected that there will be a diverse reaction in terms of complexity, dependent on the movement behaviors of the base algorithm. For example, within this study the GBFA demonstrated a reduction in complexity when compared to the base FA implementation, but both the GB and XGB frameworks showed an increase in complexity in BA and PSO, and it is expected that other nature-inspired optimization algorithms may also show an improvement in terms of complexity. An area to explore to further reduce the complexity requirements of the GB and XGB frameworks, adjustments could be made to the dynamic nature of the grouping mechanism, for example having groups stay static for t iterations after formation. For future work, the proposed algorithms will be further tested to fully investigate the impact of group and population sizes on the performance of the algorithms. The proposed algorithms can also be applied to problems other than feature selection, and the investigation of multi-objective versions of the proposed algorithm variants should also be considered.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Almomani O (2020) A feature selection model for network intrusion detection system based on PSO, Gwo, FFA and GA algorithms. Symmetry 12(6):1046. https://doi.org/10.3390/sym12061046
2. Altherwi A (2020) Application of the Firefly algorithm for optimal production and demand forecasting at selected Industrial Plant. Open J Bus Manage 08(06):2451–2459. https://doi.org/10.4236/ojbm.2020.86151
3. Alwan K, AbuEl-Atta A, Zayed H (2021) Feature selection models based on hybrid Firefly algorithm with mutation operator for network intrusion detection. Int J Intell Eng Syst 14(1):192–202. https://doi.org/10.22266/ijies2021.0228.19
4. Bacanin N et al (2023) A novel Firefly algorithm approach for efficient feature selection with covid-19 dataset. Microprocess Microsyst 98:104778. https://doi.org/10.1016/j.micpro.2023.104778
5. Cao L et al (2022) Enhancing Firefly algorithm with adaptive multi-group mechanism. Appl Intell 52(9):9795–9815. https://doi.org/10.1007/s10489-021-02766-9
6. Das S, Sahu TP, Janghel RR (2020) PSO-based group-oriented crow search algorithm (PGCSA). Eng Comput 38(2):545–571. https://doi.org/10.1108/ec-07-2019-0305
7. Hajisalem V, Babaie S (2018) A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. Comput Netw 136:37–50. https://doi.org/10.1016/j.comnet.2018.02.028
8. Hemeida AM et al (2020) Implementation of nature-inspired optimization algorithms in some data mining tasks. Ain Shams Eng J 11(2):309–318. https://doi.org/10.1016/j.asej.2019.10.003
9. Hussien AG et al (2020) Binary whale optimization algorithm for dimensionality reduction. Mathematics 8(10):1821. https://doi.org/10.3390/math8101821
10. Jain A, Sharma S, Sharma S (2021) 'Firefly algorithm', nature-inspired algorithms applications, pp 157–180. https://doi.org/10.1002/9781119681984.ch6
11. Kılıç F, Kaya Y, Yildirim S (2021) A novel multi population based particle swarm optimization for feature selection. Knowl Based Syst 219:106894. https://doi.org/10.1016/j.knosys.2021.106894
12. Li J et al (2017) Feature selection. ACM-CSUR 50(6):1–45. https://doi.org/10.1145/3136625
13. Lin X et al (2019) A new feature selection method based on symmetrical uncertainty and interaction gain. Comput Biol Chem 83:107149. https://doi.org/10.1016/j.compbiolchem.2019.107149
14. Malhotra P, Kumar D (2019) An optimized face recognition system using cuckoo search. J Intell Syst 28(2):321–332. https://doi.org/10.1515/jisys-2017-0127
15. Mohy-eddine M et al (2023) An intrusion detection model using election-based feature selection and K-NN', microprocessors and microsystems, p 104966. https://doi.org/10.1016/j.micpro.2023.104966
16. Moradi P, Gholampour M (2016) A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy. Appl Soft Comput 43:117–130. https://doi.org/10.1016/j.asoc.2016.01.044

17. Nakariyakul S (2018) High-dimensional hybrid feature selection using interaction information-guided search. Knowl Based Syst 145:59–66. https://doi.org/10.1016/j.knosys.2018.01.002
18. Robson A, Mistry K, Woo W (2023) A novel group-based Firefly algorithm with adaptive intensity behaviour. Proc 15th Int Conf Agents Artif Intell [Preprint]. https://doi.org/10.5220/0011672200003393
19. Sakri SB, Rashid A, N.B. and, Muhammad Zain Z (2018) Particle swarm optimization feature selection for breast cancer recurrence prediction. IEEE Access 6:29637–29647. https://doi.org/10.1109/access.2018.2843443
20. Seetharaman A, Sundersingh AC (2021) Gene selection and classification using correlation feature selection based binary BAT algorithm with greedy crossover. Concurrency Computation Pract Experience 34(5). https://doi.org/10.1002/cpe.6718
21. Taha AM, Mustapha A, Chen S-D (2013) Naive Bayes-guided Bat Algorithm for feature selection. Sci World J 2013:1–9. https://doi.org/10.1155/2013/325973
22. Tong N et al (2017) A multi-group Firefly algorithm for numerical optimization. J Phys Conf Ser 887(1):012060. https://doi.org/10.1088/1742-6596/887/1/012060
23. Tripathi, D. et al. (2021) Bat algorithm based feature selection: Application in credit scoring. J Intell Fuzzy Syst 41(5):5561–5570. https://doi.org/10.3233/jifs-189876
24. Wang C et al (2019) Attribute reduction based on K-Nearest neighborhood rough sets. Int J Approximate Reasoning 106:18–31. https://doi.org/10.1016/j.ijar.2018.12.013
25. Xiang J et al (2015) A novel hybrid system for feature selection based on an improved gravitational search algorithm and K-NN Method. Appl Soft Comput 31:293–307. https://doi.org/10.1016/j.asoc.2015.01.043
26. Yang XS (2011) Bat algorithm for multi-objective optimisation. Int J Bio-Inspired Comput 3(5):267. https://doi.org/10.1504/ijbic.2011.042259
27. Yang XS, He X (2013) Firefly Algorithm: recent advances and applications. Int J Swarm Intell 1(1):36. https://doi.org/10.1504/ijsi.2013.055801
28. Zhao X et al (2022) Multi-swarm improved moth–flame optimization algorithm with chaotic grouping and gaussian mutation for solving engineering optimization problems. Expert Syst Appl 204:117562. https://doi.org/10.1016/j.eswa.2022.117562