Neural Computing and Applications (2022) 34:17051–17069
https://doi.org/10.1007/s00521-022-07356-5

ORIGINAL ARTICLE

A hybrid Genetic–Grey Wolf Optimization algorithm for optimizing Takagi–Sugeno–Kang fuzzy systems

Sally M. Elghamrawy[1] · **Aboul Ella Hassanien**[2]

Received: 19 November 2021 / Accepted: 26 April 2022 / Published online: 30 May 2022

Abstract
Nature-inspired optimization techniques have been applied in various fields of study to solve optimization problems. Since designing a Fuzzy System (FS) can be considered one of the most complex optimization problems, many meta-heuristic optimizations have been developed to design FS structures. This paper aims to design a Takagi–Sugeno–Kang fuzzy Systems (TSK-FS) structure by generating the required fuzzy rules and selecting the most influential parameters for these rules. In this context, a new hybrid nature-inspired algorithm is proposed, namely Genetic–Grey Wolf Optimization (GGWO) algorithm, to optimize TSK-FSs. In GGWO, a hybridization of the genetic algorithm (GA) and the grey wolf optimizer (GWO) is applied to overcome the premature convergence and poor solution exploitation of the standard GWO. Using genetic crossover and mutation operators accelerates the exploration process and efficiently reaches the best solution (rule generation) within a reasonable time. The proposed GGWO is tested on several benchmark functions compared with other nature-inspired optimization algorithms. The result of simulations applied to the fuzzy control of nonlinear plants shows the superiority of GGWO in designing TSK-FSs with high accuracy compared with different optimization algorithms in terms of Root Mean Squared Error (RMSE) and computational time.

Keywords Nature-inspired optimization methods · Takagi–Sugeno–Kang Fuzzy System · Grey Wolf Optimizer (GWO) · Fuzzy rules · Genetic algorithm (GA)

1 Introduction

Fuzzy systems have been extensively applied in multiple areas, including decision analysis, automatic control, and system modeling [1]. The membership functions are responsible for mapping the input variables of the fuzzy system. These mappings are then inputted into the fuzzy rules. The generated fuzzy rules are responsible for deciding which action to take. Consequently, designing a fuzzy system structure is one of the most critical processes in FS applications and can be considered a complex optimization problem. Generating the fuzzy rules in a Takagi–Sugeno–Kang Fuzzy System (TSK-FS) and selecting the parameters in these rules are considered an optimization problem that enhances system accuracy. However, fuzzy rule generation is problematic and time-consuming. Therefore, the traditional computation and statistical approaches [2] for automatically generating fuzzy rules cannot enhance the FS performance.

Nature-inspired optimization methods provide an efficient way to find the optimal solution in complex real-world problems [3], such as the optimal design of fuzzy systems and classification problems. Nature-inspired optimization algorithms can be inspired by animal behaviors or evolutionary concepts based on individuals' populations.

Extraordinary efforts have been made to optimize the design of TSK-FSs using evolutionary algorithms (EAs) [4], various genetic algorithms (GAs) [5–7], and nature-inspired optimization algorithms such as particle swarm optimization (PSO) [8–10], ant colony optimization (ACO)

✉ Aboul Ella Hassanien
Aboitcairo@cu.edu.eg

Sally M. Elghamrawy
sally_elghamrawy@ieee.org; sally@mans.edu.eg

[1] MISR Higher Institute for Engineering and Technology, Mansoura & Scientific Research Group in Egypt (SRGE), Egypt

[2] Faculty of Computers & AI, Cairo University & Scientific Research Group in Egypt (SRGE), Cairo, Egypt

[11], the gravitational search algorithm (GSA) [12], and the grey wolf optimizer (GWO) [13–15].

GWO is a relatively new nature-inspired optimization algorithm that mimics grey wolves' social hierarchy and their hunting mechanisms in the wild. The main advantage of GWO lies in its reduced number of search parameters, which results in its superiority on different optimization problems, including optimal tuning of PID-fuzzy controllers [16], training multi-layer perceptrons [17], and optimization benchmarks [18].

This advantage has motivated several researchers [19–24] to hybridize two or more meta-heuristics to improve the performance of GWO for optimizing FSs. The authors of [19] recently proposed gDE-GWO, which is a hybridization of GWO and differential evolution (DE) mutation [20]. In [21], the GWO was improved by altering the equation for finding the following positions of the wolves; then, the IGWO with Fuzzy PID controller was applied to power system load frequency control (LFC). Tawhid et al. [22] proposed the hybrid algorithm HGWOGA, which combined the GA and GWO algorithms to minimize a simplified model of a molecule's energy function. A modified version of the GWO algorithm was proposed in [23], namely, the MGWO-based cascade PI-PD controller; the authors assigned more importance to the alpha wolves to find the optimal solution during the iterations.

This paper aims to introduce a hybrid Genetic–Grey Wolf Optimization (GGWO) algorithm for TSK-FS optimization through five main stages. The contributions of the proposed GGWO lie in two main directions.

First, the standard GWO is improved [13] by embedding the GA's crossover and mutation operations to overcome the premature convergence of poor exploitation of solutions in the standard GWO. It will help optimize the search process to find the next locations for the wolves and optimal solutions during iteration. Also, this hybridization accelerates the exploration process and reaches the best solution in a reasonable time.

Second, because designing a fuzzy system can be considered an optimization problem, GGWO is used to optimize TSK-FSs by designing their structure, using rule generation, and optimizing the parameters for each fuzzy rule in the TSK-FSs. The proposed algorithm shows promising performance in optimizing FS compared to other optimization algorithms, as demonstrated in Sect. 5.

The main contribution is summarized as follows.

- A new hybrid nature-inspired algorithm is proposed, namely Genetic–Grey Wolf Optimization (GGWO) algorithm, to optimize TSK-FSs.
- GGWO is applied to overcome the premature convergence and poor solution exploitation of the standard GWO.
- Using genetic crossover and mutation operators accelerates the exploration process and efficiently reaches the best solution (rule generation) within a reasonable time.
- The proposed GGWO is tested on several benchmark functions compared with other nature-inspired optimization algorithms.
- The result of simulations applied to the fuzzy control of nonlinear plants shows the superiority of GGWO in designing TSK-FSs with high accuracy compared with different optimization algorithms.

This paper's remainder is organized as follows. Section 2 describes related studies in fuzzy system optimization and introduces the GWO. In Sect. 3, the first-order TSK-FS is defined as a model that can be formulated as an optimization problem; its main goal is to find the optimal convenient solution. Section 4 describes the proposed GGWO. Section 5 presents GGWO simulation results compared to other optimization algorithms, and Sect. 6 concludes the paper.

2 Related work

When designing the structure of TSK-FSs, there are several considerations, including accuracy and interpretability [25]. To preserve a fuzzy model's accuracy, the number of fuzzy rules must be determined, and the parameters in each fuzzy rule must be selected appropriately. Also, constraints that optimize the interpretability of TSK-FSs must be considered. Several studies have been proposed to optimize the design process for TSK-FS models. In [26], generating fuzzy rules was presented based on a fuzzy genetic system. In [10], a constrained PSO algorithm (C-PSO) was proposed to set linear constraints to enhance interpretability while preserving a TSK-FS model's accuracy. Juang et al. [9] proposed a hierarchical cluster-based multispecies particle swarm optimization (HCMSPSO) algorithm for TSK-FS optimization; the authors designed both the structure (the number of rules) and the parameters of an FS. In [27], the authors proposed a new approach for generating accurate and interpretable fuzzy rules: using only fuzzy labels to build transparent knowledge-based systems. In [28], a fuzzy genetic programming approach called FGPRL was proposed to determine the fuzzy rules' size and adjust the controller parameters based on reinforcement learning.

Recently, many researchers [29–34] have investigated the optimization algorithms in TSK-FS models and fuzzy rules as follows: Deng et. al. [34] investigated a monotonic relation between the inputs and outputs of the TSK-FS model and presented a Tikhonov regularization strategy to avoid the loss. In [29], the authors used sparse regressions to train high-order TSK-FS models. The sparse regressions used offer regularization; it means that it can be used when the variables number increases the observations. Adequate monotonicity conditions for zero-order TSK-FS models with membership functions generated by cubic splines are proposed in [35], the model concerned with the optimization problems.

Wu et al. [36] proposed three influential optimization techniques based on neural networks for training TSK-FS models: the MBGD with regularization, DropRule, and AdaBound (MBGD-RDA). In [37], the authors improved the MBGD-RDA proposed in [36] by using fuzzy c-means clustering and AdaBound, in a new algorithm named FCM-RDpA, to optimize the parameters used in generating the rules. Wang et al. [38] proposed an aggregation method using fuzzy neural networks for TSK-FS models, namely TSKFNN. The authors implemented AdaBoost to gain a combination of trusty rules to generate prediction models using a number of real datasets.

In [39], the authors proposed a graph-based TSK-FS model to predict hemodialysis patients' adequacy.

2.1 Grey Wolf Optimizer (GWO)

The GWO is a nature-inspired optimization algorithm that Seyedali Mirjalili first proposed in 2014 [13]. The GWO algorithm mimics the social hierarchy of grey wolves and their hunting mechanisms in the wild. The wolves' hunting mechanisms are used to represent a path to the solution of the optimization problem.

Grey wolves tend to live in packs. To simulate the wolf social hierarchy, suppose that there are each pack of wolves contains four different types of wolves: alpha (α), beta (β), delta (δ), and omega (ω); thus, the wolves can be categorized into four classes. The population of search agents (wolves) is categorized according to their fitness function. This hierarchy's mathematical model is designed by considering the wolf with the best fitness function as the alpha wolf (The leader). The second-best fitness is the beta wolf, and the third-best solution is the delta wolf. Any other solution will be considered the omega wolves (the lowest-ranked wolves), who will follow the other three types. This hierarchy is updated in every iteration to change the solutions. The hunting mechanisms of the grey wolves are mathematically modeled via the following steps.

2.1.1 Searching for the prey

The search process (exploration) is implemented based on α, β, and δ. Their positions are varied randomly to foster a global search for prey. The coefficient vectors $\vec{A}$ are assigned random values ranging from $-1 > \overrightarrow{A} > 1$, and the coefficient vectors $\vec{C}$ are assigned random values from [0 to 2], as shown in Fig. 1a, to enable the wolves to find more suitable prey.

2.1.2 Encircling prey

Grey wolves encircle their prey during a hunt; the following equations simulate the encircling process:

$$\vec{D} = \left|\vec{C}.\vec{V}_{p(i)} - \vec{V}_{(i)}\right| \tag{1}$$

$$\vec{V}_{(i+1)} = \vec{V}_{p(i)} - \overrightarrow{A} \cdot \overrightarrow{D} \tag{2}$$

where $\vec{V}_p$ is the position vector of the prey, i indexes the current iteration, and $\vec{V}$ indicates the position vector of a grey wolf. $\overrightarrow{A}$ and $\vec{C}$ are calculated as follows:

$$\vec{\mathrm{A}} = 2\vec{\mathrm{a}}.\vec{\mathrm{r}}_1 - \vec{\mathrm{a}} \tag{3}$$

$$\vec{C} = 2.\overrightarrow{r}_2 \tag{4}$$

During the iterations, the values of $\vec{\mathrm{a}}$ vary from 0 to 2 and $\vec{\mathrm{r}}_1$ and $\overrightarrow{r}_2$ are assigned random values from 0 to 1 to allow the wolves to update their positions around the prey's location using (1) and (2), as shown in Fig. 1b. In Fig. 1b, suppose that the location of a grey wolf is (X, Y) and the location of the prey is (X',Y'): the wolves can then update their positions based on prey's location.

2.1.3 Hunting the prey

The alpha wolf is responsible for the hunting process [40] (exploitation), because in most cases, the α wolf leads the pack and can detect the location of prey. However, to

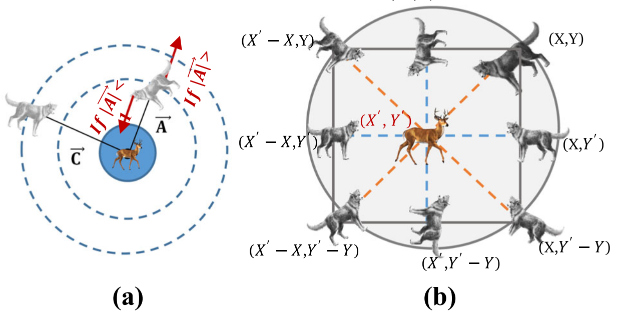


Fig. 1 **a** Effect of the coefficient vectors A→ and C→and **b** Position vectors of the grey wolves around the prey

mathematically model the optimum location of the prey, the first three best solutions are obtained from the alpha (α), beta (β), and delta (δ) wolves and are calculated as follows:

$$\vec{D}_\alpha = \left|\overrightarrow{C_1}.\vec{V}_\alpha - \vec{X}\right|$$
$$\vec{D}_\beta = \left|\overrightarrow{C_2}.\vec{V}_\beta - \vec{X}\right| \quad (5)$$
$$\vec{D}_\delta = \left|\overrightarrow{C_3}.\vec{V}_\delta - \vec{X}\right|$$

$$\vec{V}_1 = \vec{V}_\alpha - \overrightarrow{A_1}\cdot\left(\overrightarrow{D_\alpha}\right)$$
$$\vec{V}_2 = \vec{V}_\beta - \overrightarrow{A_2}\cdot\left(\overrightarrow{D_\beta}\right) \quad (6)$$
$$\vec{V}_3 = \vec{V}_\delta - \overrightarrow{A_3}\cdot\left(\overrightarrow{D_\delta}\right)$$

Then, the other agents (wolves) must update their positions based on the positions of the three best search agents (α, β, δ) as shown in the following equation:

$$\vec{V}_{(i+1)} = \frac{\vec{V}_1 + \vec{V}_2 + \vec{V}_3}{3} \quad (7)$$

Thus, the α, β, and δ wolves determine the prey's position, and the other wolves sequentially update their positions around the prey.

2.1.4 Attacking the prey

When the wolves surround the prey, it stops moving. Then, the grey wolves attack the prey to finish the hunting process. To simulate the attacking process, the value of $\vec{a}$ is decreased; as a result, the value of $\overrightarrow{\mathrm{A}}$ also decreases. The result of the attack process is shown in Fig. 2. $\overrightarrow{\mathrm{A}}$ is considered as having a value from 2a to -2a. The next position of the grey wolf will be between its current location and the location of the prey if the value of $\overrightarrow{\mathrm{A}}$ is between 1 and $-$ 1.

3 Problem statement

This section describes the first order TSK-FS [41] to be optimized and shows the mathematical functions for the TSK-FS model; these include the fuzzy rules in the following form:

$$\mathbb{R}_j : Ifx_1(m) \text{ is } A_{j1},\ \text{and}\ldots \text{ and } \ x_p(m) \text{ is } A_{jp} \quad (8)$$

THEN $yis\mathbb{Z}_j(x) = a_{j0} + a_{j1}x_1 + \ldots + a_{jp}x_p$. where $\mathbb{R}_j$ is the j^{th} rule, $\in \mathbb{R}^n$, A_{jp} are the fuzzy sets, $y \in \mathbb{R}$, and j ranges from [1 to c]:

$$A_{jp}(x_p) = \exp\left(\left(\frac{X_p - V_{jp}}{\sigma_{jp}}\right)^2\right) \quad (9)$$

where V_{jp} and σ_{jp} are the parameters of the fuzzy set, and $a_{jp} \in A_{jp}$, where p ranges from [1 to IS] and IS is the dimension of the input space.

Suppose a dataset $x = \{x_1, \ldots\ldots x_n\}$. The strength of fuzzy rule j is calculated as follows:

$$\Phi_j(\vec{x}) = \prod_{i=1}^{n} \mathcal{M}_{ji}(x_i) \quad (10)$$

$$\Phi_j(\vec{x}) = \exp\left\{-\sum_{i=1}^{n}\left(\frac{X_p - V_{jp}}{\sigma_{jp}}\right)^2\right\} \quad (11)$$

Gaussian fuzzy sets represent local information efficiently and contain continuous derivatives of the values. In first-order TSK FSs, the consequent function $\mathbb{Z}_j(x)$ is set to a linear function of the input variables. The output of the system is given by:

$$\hat{y} = \sum\nolimits_{j=1}^{c}\left[\prod\nolimits_{P=1}^{IS} A_{jp}(x_p)\mathbb{Z}_j(x)\right]\Big/\sum\nolimits_{k=1}^{c}\left[\prod\nolimits_{P=1}^{IS} A_{kp}(x_p)\right] \quad (12)$$

where c is the number of rules in the FS. The proposed GGWO algorithm's main goal is to optimize the parameters used in each fuzzy rule and specify the number of rules c.

4 The proposed hybrid Genetic–Grey Wolf optimizer (GGWO) algorithm

The proposed GGWO algorithm optimizes the fuzzy system in the five main stages shown in Fig. 2. (1) initial population, (2) fitness function evaluation stage, (3) exploration stage, (4) exploitation stage, and (5) termination stage.

For the GGWO algorithm, a hybrid combination of the GA [42] and the GWO [13] is proposed in this paper. This hybridization aims to overcome the premature convergence and the poor exploitation of solutions of the standard GWO algorithm by using crossover and mutation operations during the exploitation and exploration stages of the GWO algorithm. It will optimize the process of searching for the optimal solutions. The GGWO operating sequence starts by randomly generating the search agents (wolves) that form the pack of grey wolves. Each wolf in the pack is assigned to a position in the vector $\overrightarrow{V_i(m)}$ as follows:

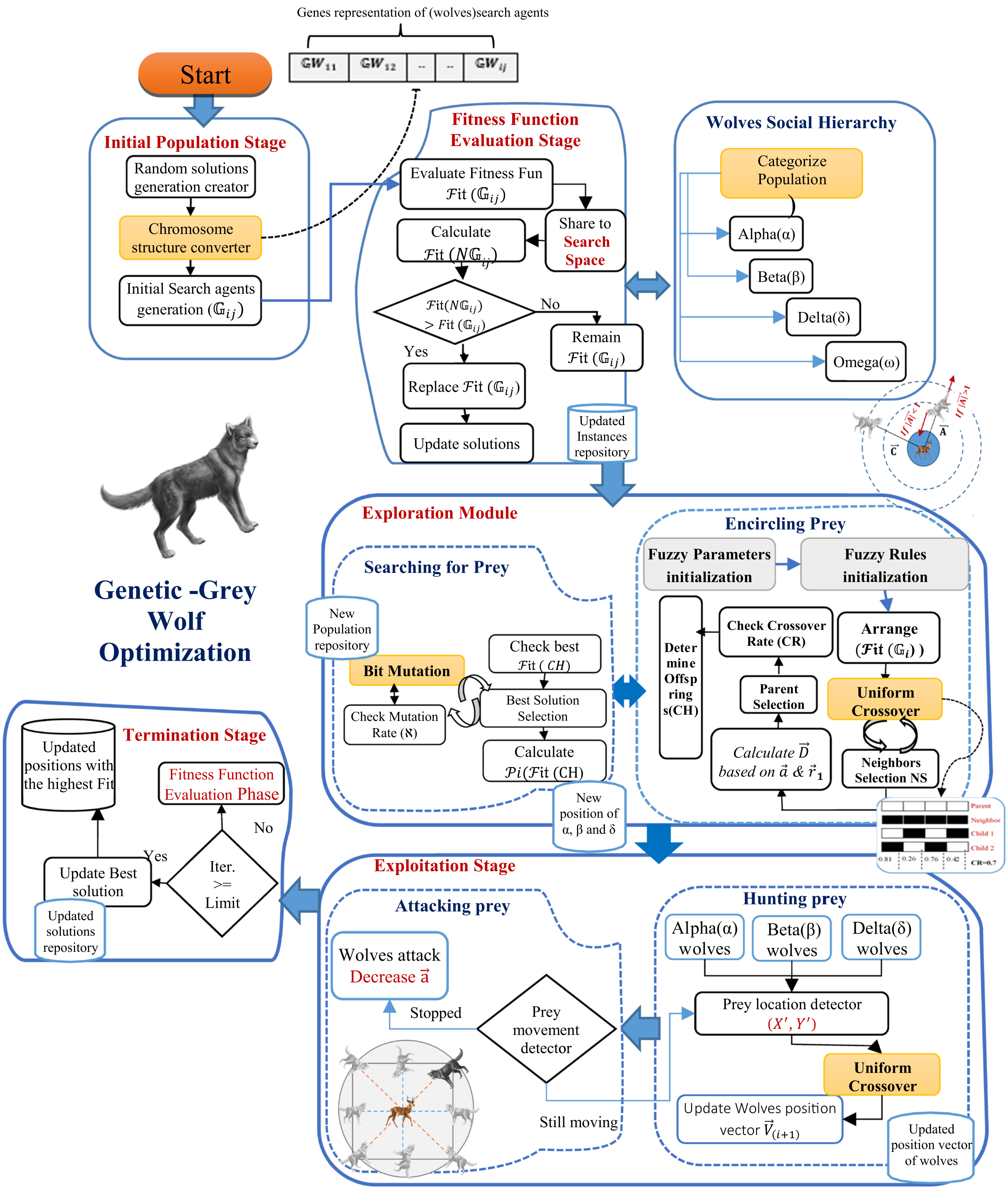


Fig. 2 Proposed Genetic Grey Wolf Optimization (GGWO) algorithm

$$\overrightarrow{V_i(x)} = [v_i^1(x)..v_i^2(x)..v_i^{curnt^i}(x)]^T \quad i = 1, 2\ldots.N \tag{13}$$

where N is the total number of search agents (wolves), $v_i^{curnt^i}(x)$ is the position of the i^{th} wolf in the pack, $curnt^i$ is the index of the current iteration, and x ranges from 1 to a predefined maximum number of iterations, x_{max}.

Several chromosomes represent these solutions. Partitioning a solution into several chromosomes allows diversity during the wolves' search process for the optimal solution. In the initial population stage, each solution is represented by several genes.

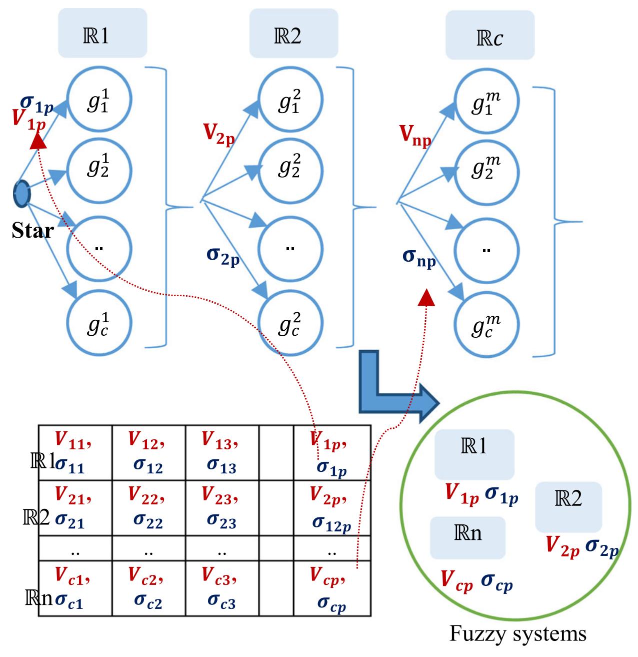
$\mathbb{G}W_{ij}$. An initial process to generate the random solutions $\mathbb{G}w_{ij}$ is applied based on the upper and lower boundary, as shown in Eq. (14).

$$\mathbb{G}W_{ij} = \mathbb{G}W_{LOwj} + z \times \left(\mathbb{G}W_{Upj} - \mathbb{G}_{WLOwj}\right) \tag{14}$$

where is a random value ranging from 0 to 1, $i = 1, 2\ldots \mathcal{P}_{Size}$, $\mathcal{P}_{Size}$ is the population size (based on the number of wolves), and $j = 1, 2, \ldots. g_n$.

Based on the problem's fitness function, the chromosomes with the highest fitness values will gradually supplant the chromosomes with the lowest fitness values.

This paper calculates the fitness function in terms of the RMSE between the desired and actual outputs. In the proposed GGWO, the number of rules c in the fuzzy system equals the number of genes g_n represented, as shown in Fig. 3.



ℝ1
ℝ2
ℝc
σ_{1p}
V_{1p}
Star
g_1^1
g_2^1
..
g_c^1
V_{2p}
σ_{2p}
g_1^2
g_2^2
..
g_c^2
V_{np}
σ_{np}
g_1^m
g_2^m
..
g_c^m
ℝ1 V_{11}, σ_{11} | V_{12}, σ_{12} | V_{13}, σ_{13} | | V_{1p}, σ_{1p}
ℝ2 V_{21}, σ_{21} | V_{22}, σ_{22} | V_{23}, σ_{23} | | V_{2p}, σ_{12p}
.. | .. | .. | | ..
ℝn V_{c1}, σ_{c1} | V_{c2}, σ_{c2} | V_{c3}, σ_{c3} | | V_{cp}, σ_{cp}
ℝ1
V_{1p} σ_{1p}
ℝ2
ℝn
V_{2p} σ_{2p}
V_{cp} σ_{cp}
Fuzzy systems

Fig. 3 The fuzzy rule representations with their corresponding parameters

The reason is that each rule has several parameters to be optimized that cannot be divided into different genes. Consequently, the fitness function for a fuzzy rule directly influences the overall performance of the FS. Suppose that each parameter of the jth rule in (8) is optimized by the jth gene; then, these parameters are denoted as the elements of a gene position vector $\overrightarrow{\mathbb{G}W_j}$ for the first-order TSK FSs, as follows:

$$\overrightarrow{\mathbb{G}W_j} = [V_{j1}, \sigma_{j1}, \ldots, V_{j1}, \sigma_{j1,}a_{j0}, a_{j1}, \ldots, a_{jn}] \tag{15}$$

The overall performance of the FS is directly affected by the genes' fitness in the fuzzy results. In the fitness function evaluation stage, each gene's fitness function is calculated in the fuzzy rules during each iteration to find the optimal solution that represents the highest fitness solution obtained.

The search process terminates when it reaches a predefined maximum number of iterations (x_{max}). Several genes represent each solution (grey wolf). To determine whether a new fuzzy rule should be generated, the rule strength $\Phi_j(\vec{x})$ is calculated by (11) for each upcoming data item $\vec{x}(k)$:

$$\rho = Max_i\{\Phi_j(\vec{x}(k))i = 1, \ldots.c \tag{16}$$

where c is the number of rules. The rule with the maximum strength (whose $\Phi_j \le \Phi_{ThrS}$) will be generated, where $\Phi_{ThrS} \in [0, 1]$ is a predefined threshold.

In GGWO, the newly generated rule will cause new genes to be developed. When a determination is made that a fuzzy rule should be generated, the centre and width of the Gaussian fuzzy set are denoted by:

$$V_{j1} = x_i(0) \quad \text{and} \quad \sigma_{j1} = \sigma_{initW} \quad i = 1, 2, \ldots n \tag{17}$$

where σ_{initW} is the initial width of the fuzzy set, and

$$V_{(c+1)i} = x_i(k) \quad i = 1, 2, \ldots n \tag{18}$$

$$\sigma_{(c+1)i} = \Theta\left\{ - \sum_{j=1}^{n} \left(\frac{x_{ip}(k) - v_{ip}}{\sigma_{ip}} \right)^2 \right\} \tag{19}$$

where Θ is the overlap degree between two rules ($\Theta > 0$ and is a predefined value). Based on (18) and (19), the GGWO algorithm's genes will be initialized for each rule. Selecting the fitness function is the most crucial step in the optimization process: it is the guide for evaluating the population's solutions. The fitness value for each gene is calculated as follows:

$$\mathcal{F}\text{it}(\mathbb{G}w_i) = \sum_{i=1}^{g_n} \frac{1}{(y_i - \hat{y}_i)^2} \tag{20}$$

The fitness values are assigned different weights. Then, the fitness function is evaluated for each solution, as shown below:

$$\mathcal{F}\text{it}\left(\mathbb{G}w_{ij}\right) = \sum_{i=1}^{\mathcal{P}_{Size}} \sum_{j=1}^{g_n} \frac{1}{\left(y_i - \hat{y}_{\boldsymbol{i}}\right)_j^2} \tag{21}$$

The search agents (wolves) share the fitness value assigned to the search space. The search agents return to the original solutions and choose new solutions, after which their new fitness values $\mathcal{F}\text{it}(N\mathbb{G}\text{w}_{ij})$ are evaluated. Then, these new fitness values are compared with the previous values. When the new values are greater than the old ones, the search agents replace the old values with the new solution values. Otherwise, they retain the old solutions.

In the *wolf social hierarchy sub-stage*, the solutions are classified into different classes based on their fitness functions, as shown in Fig. 4.

The solution with the highest fitness function is the alpha wolf (the leader), the solution with the second-best fitness is the beta wolf, and the one with the third-best fitness is the delta wolf. All other solutions are considered omega wolves (the lowest-ranked wolves), following the three different types. Each iteration of the search process consists of two main stages: an exploration stage and an exploitation stage. In the exploration stage, the wolves' main goal is to search and encircle the prey.

Fig. 4 Proposed fitness evaluation algorithm (social hierarchy)

Algorithm 1: Fitness Evaluation algorithm (Social hierarchy)

```
$\mathcal{P}_{Size}$: population size is the number of wolves which are initialized randomly in the search space
$g_n$: number of genes
$Pos_i$: Position of search agents
$x_{max}$ . Maximum number of iterations
Step 1: Randomly initialize population
 For each iter in $x_{itr}$
     aa: Initial Population= {$\mathbb{G}w_{11}$, $\mathbb{G}w_{12}$, $\mathbb{G}w_{13}$ .... $\mathcal{P}_{Size}$ };
Step 2: Calculate Fitness Function
        For j =1 to $g_n$
            Calculate Fitness Fun $\mathcal{F}it$ ($\mathbb{G}w_{ij}$) ;
            Share to Search Space ;
        End For
     End for
            Return to Population ;
Step 3: Social hierarchy of wolves
        For I =1 to $\mathcal{P}_{Size}$
          For j =1 to $g_n$
            Calculate new $\mathcal{F}it$ (N$\mathbb{G}_{wij}$) ;
            if $\mathcal{F}it$(N$\mathbb{G}_{ij}$) > $Fit$ ($\mathbb{G}w_{ij}$) then
             Replace $\mathcal{F}it$ ($\mathbb{G}w_{ij}$) ;
            Update solutions ;
            Else
            Remain $\mathcal{F}it$ ($\mathbb{G}w_{ij}$) ;
            αw ← Max ($\mathcal{F}it$ ($\mathbb{G}w_{ij}$) ) ;
            βw ← Second (($\mathcal{F}it$ ($\mathbb{G}w_{ij}$));
            δw ← Third (($\mathcal{F}it$ ($\mathbb{G}w_{ij}$));
            ωw ←  Fourth(($\mathcal{F}it$ ($\mathbb{G}w_{ij}$));
     Decompose the $\rho_{best}$ into the fuzzy set centers and widths
          End For
        End for
Step 4: Termination check
        If  $itr$ > $x_{max}$

           Update solutions;
          Solution positions repository;
          Else go to aa;
        End if
End for
```

In the *exploration stage* of the proposed GGWO, the GA processes of crossover and mutation are applied to each population to increase the search's diversity and to overcome the premature convergence of the standard GWO algorithm and its poor exploitation of solutions. In the standard GWO algorithm, a random variable $\vec{a}$ is varied from [0 to 2] to determine the prey's location and the alpha wolves.

As shown in Fig. 2, during the prey encirclement sub-stage, the parent solution P_s is assigned to the highest fitness values using (19) and the Neighbour solution N_s is assigned randomly using roulette wheel selection [43]. Then, a crossover is conducted between N_s and P_s. The N_s and P_s. solutions are combined to produce one or more children, based on the value of a crossover rate (CR), which is predefined parameter. To determine the nearest solution to the parent, GGWO calculates the value of $\vec{D}$ based on $\vec{a}$. and $\vec{r}_1$ using N_s, as own in Fig. 5.

The mutation operation is used during the search process for prey (solutions) as defined in the following algorithm:

```
Algorithm 2: Mutation (p1, p2)
1.  Randomly set a number for the mutation rate $\aleph \in (0, 1)$
2.  Calculate $\sum_{i=1}^{ch} \mathcal{F}\mathrm{it}_i(\mathrm{Ch}_s{}^i)$
3.  If $\mathcal{F}\mathrm{it}_i(\mathrm{Ch}_s{}^i) > \mathcal{F}\mathrm{it}_{i+1}(\mathrm{Ch}_s{}^{i+1})$
4.  Then, $\boldsymbol{New}\mathbb{G}_i \leftarrow \mathcal{F}\mathrm{it}_i(\mathrm{Ch}_s{}^i)$
5.  If $\boldsymbol{New}\mathbb{G}_i < \aleph$ then
6.  $New_{ij} = PS_{ij} + \boldsymbol{\alpha}_{\mathbf{ij}}\,(RS_{ij} - PS_{ij})$
```

where New_{ij} is the new solution generated, PS_{ij} is the parent solution, $\boldsymbol{\alpha}_{\mathbf{ij}}$ is a random number in the range [− 1,1], and RS_{ij} is the random solution selected.

In the *exploitation stage*, the top three wolves (α, β, and δ) determine the optimum prey location. The top three solutions obtained are *calculated* as follows:

$$v_i^S(x) = \left[v_1^{S_i}(x)\ldots v_2^{S_i}(x)\ldots v_i^{S_i}(x)\right], S \in \{\alpha, \beta, \delta\} \tag{22}$$

The selection process for the three-vector solutions are calculated as follows:

$$\mathcal{F}\mathrm{it}_\alpha\left(v^\alpha(x)\right) = \mathrm{Max}_{i=1..n}\mathcal{F}\mathrm{it}_{\alpha i}\left(v_i^\alpha(x)\right)|v^\alpha(x) \in V_i(x) \tag{23}$$

$$\mathcal{F}\mathrm{it}_\beta\left(v^\beta(x)\right) = Max_{i=1..n}\mathcal{F}\mathrm{it}_{\beta i}\left(v_i^\beta(x)\right)|v^\beta(x) \in V_i(x)\{v^\alpha(x)\} \tag{24}$$

$$\mathcal{F}\mathrm{it}_\delta\left(v^\delta(x)\right) = Max_{i=1..n}\mathcal{F}\mathrm{it}_{\delta i}\left(v_i^\delta(x)\right)|v^\delta(x) \in V_i(x)\backslash\{v^\alpha(x), v^\beta(x)\} \tag{25}$$

where the following condition must be applied:

$$\mathcal{F}\mathrm{it}_\alpha\left(v^\alpha(x)\right) > \mathcal{F}\mathrm{it}_\beta\left(v^\beta(x)\right) > \mathcal{F}\mathrm{it}_\delta\left(v^\delta(x)\right) \tag{26}$$

A uniform crossover operation is applied between the three vectors to select and update their positions and the prey's position, as shown in Fig. 2. Finally, the best three wolves make the other wolves (omega ω) update their positions according to the best three wolves' positions.

After attacking the prey with several wolves, the GGWO checks the mobility of the prey. When the agents (wolves) confirm that the prey is at location (X',Y'), the leader agent decreases the value of $\vec{a}$, which leads to a decrease in $\vec{A}$, to begin the attacking process.

5 Experimental evaluation

This study evaluated the proposed GGWO algorithm's effectiveness through several simulated experiments and analytical results, as shown in the following subsections. The algorithms were applied on an Intel(R) Core (TM) i7 CPU with 16 GB RAM and 2.81 GHz clock speed using MATLAB R2019a.

The parameters used in the experiments to evaluate the proposed GGWO algorithm can be categorized into common parameters, which are the GA and GWO's parameters defined to achieve a balance between convergence and number of genes\solutions.

The main goal of the GGWO algorithm is to optimize the parameters of the fuzzy rules generated for TSK-FSs. These parameters are problem dependent like Θ: the overlap degree between two rules, $\Phi_{ThrS}, \sigma_{initW}$: the thresholds used, Φ_j : the maximum strength. These parameters are optimized during the implementation, and experiment five shows the influence of changing these parameters on the performance of the algorithms.

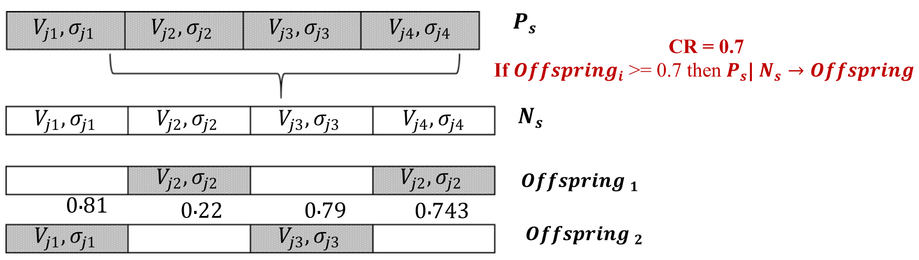


Fig. 5 Crossover operation on the (GGWO) algorithm

Table 1 GGWO parameters

Parameter	Value	Parameter	Value
Number of generations (cycle)	500	Crossover rate CR	0.7
Population size $\mathcal{P}_{\mathcal{S}ize}$	30	Mutation Rate $\aleph$	0.01
Maximum number of iterations $\mathrm{Max}_{\mathrm{it}}$	500	Number of genes/solutions g_n	6

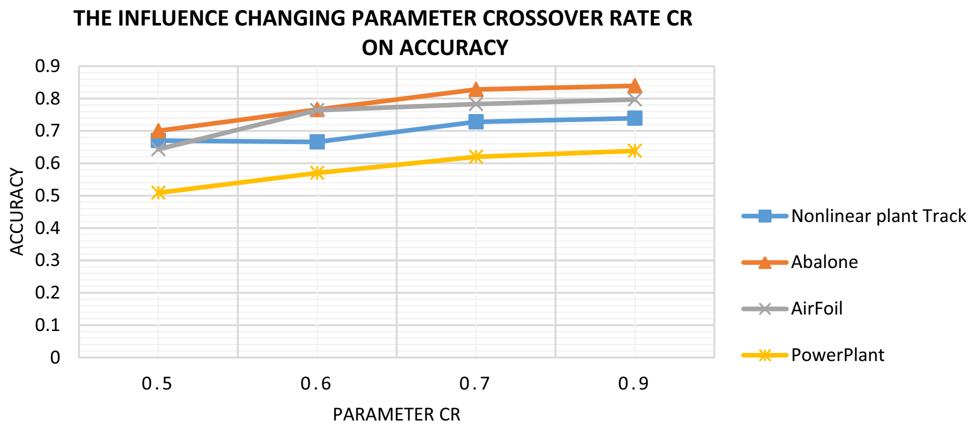


Fig. 6 Influence changing parameter crossover rate CR on the GGWO accuracy in different problems

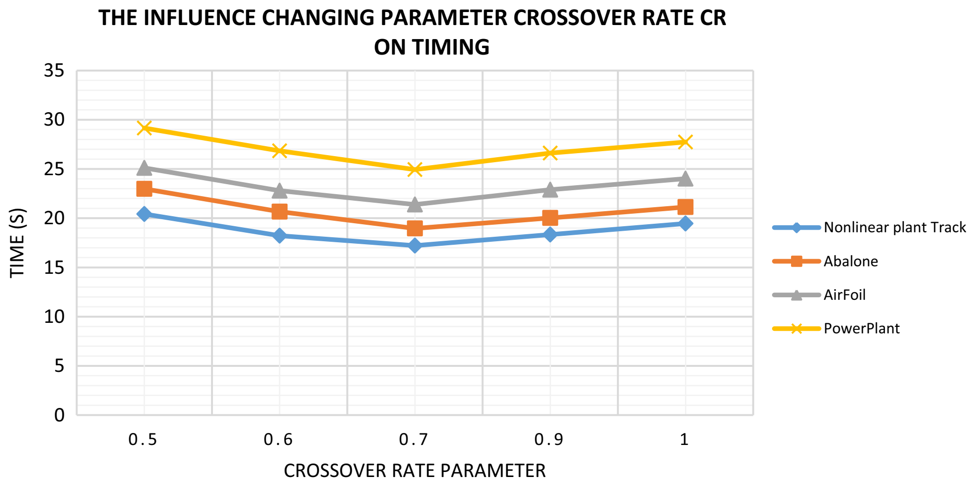


Fig. 7 Influence changing parameter crossover rate CR on the GGWO Timing in different problems

To guarantee the fairness of the comparative results, the values of the parameters used in all algorithms are set the same as follows: The maximum number of iterations is 500; the population size is 30. The crossover and mutation rate is 0.7 and 0.01, respectively. The main parameters are set as shown in Table 1.

In addition, an experiment is implemented to test the influence of changing the parameter crossover rate on the accuracy and time of classification in different problems used through the evaluation, as shown in Figs. 6 and 7.

As shown from Figs. 6 and 7, it is concluded that choosing a CR rate of more than 0.6 improves the classification accuracy, but it starts to be stable in accuracy degree when CR's value is from 0.7 to 0.9. However, increasing simulation time is shown when the CR exceeds 0.7. The results displayed in the figures confirm that the value of 0.7 seems to be an acceptable comprise between accuracy and timing, as this value is adapted in the most published work.

Table 2 Benchmark functions

Unimodal test functions	Range
$F_1(X) = \sum_{i=1}^{n} x_i^2$	[− 100,100]
$F_2(X) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[− 10,10]
$F_3(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	[− 100,100]
$F_4(X) = max_i\{\lvert x_i \rvert, 1 \leq i \leq n\}$	[− 100,100]
$F_5(X) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	30,30]

Multimodal test functions	Range
$F_8(X) = \sum_{i=1}^{n} -x_i Sin + \sqrt{\lvert x_i \rvert}$	[− 500, 500]
$F_9(X) = \sum_{i=1}^{n} [x_i^2 - 10Cos(2\pi x_i) + 10]$	[− 5.12, 5.12]
$F_{10}(X) = -20 \exp\left[-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right] - \exp\left[\frac{1}{n}\sum_{i=1}^{n} Cos(2\pi x_i)\right] + 20 + e$	[− 32, 32]
$F_{11}(X) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[− 600, 600]

5.1 Experiment one: test the efficiency of the proposed GGWO using benchmark functions

The performance of the GGWO algorithm is evaluated using some standard benchmark functions [13]. In this experiment, the GGWO algorithm is tested using nine benchmark functions due to space limitations. F_1, F_2, F_3 , F_4, and F_5 are unimodal functions while F_8, F_9, F_{10}, and F_{11} are multimodal functions. These functions were selected both for their simplicity and to allow GGWO's results to be compared with other algorithms' results. The nine functions are listed in Table 2. The best solution (F_{min}) obtained for F_8 is 418.9829 $\times$ 5, while the F_{min} for the other functions is zero. The dimensions of the nine functions are 30, and the boundary of the search space (range) of each function is listed in Table 2.

The average (Avg) and standard deviation (StD) are calculated for each function using thirty runs, as shown in Table 3. To verify the obtained results, GGWO is compared with Original GWO [13], IGWO [21], PSO [8], GSA [12], and MGWO [23].

Results' analysis for experiment one: The exploitation and exploration stages of GGWO are tested using unimodal and multimodal functions, respectively. As shown in Table 3, the proposed GGWO algorithm outperforms all the other algorithms on the unimodal functions f1, f2, f3, and f4, reflecting the superiority of GGWO in the exploitation stage. In function f5, the average value obtained by GGWO is better than other algorithms, while the standard deviation obtained by IGWO is better than GGWO. The superiority of GGWO's results occurs from the usage of the GA operators (crossover and mutation) in the exploitation stage in the GGWO algorithm. In addition, GWOA achieves competitive results on the multimodal benchmark functions compared to GWO [13], IGWO [21], PSO [8], GSA [12], and MGWO [23]. Most of the competitive algorithms suffer from becoming trapped in local optima; in contrast, the combination of GA and GWO in the proposed GGWO enhances the search process during the exploration stage and makes the algorithm reach the optimal solution within a reasonable time compared to GWO [13], IGWO [21], PSO [8], MGWO [23], FCM-RDpA [37], MBGD-RDA [36], and TSKFNN [38] algorithms as shown in Fig. 8.

From the results obtained in Fig. 8, it is found that the simulation time of the GGWO algorithm is less than that of the GWO algorithm due to the crossover and mutation operation implemented in GWO, which determines the position of a Grey wolf faster.

5.2 Experiment two: test the efficiency of the proposed GGWO in TSK-FS optimization

This experiment uses an optimization example to test the performance of GGWO. The results obtained are compared with those obtained by other optimization algorithms using the same examples. The RMSE is the metric used to

Table 3 Results of the benchmark functions by GGWO compared with other algorithms

	GGWO		GWO		IGWO		PSO		GSA		MGWO	
	Avg.	StD.	Avg.	StD.	Avg.	StD.	Avg.	StD.	Avg.	StD.	Avg.	StD.
	Unimodal test functions											
F1	**1.20E − 39**	**6.00E − 20**	6.59E − 28	6.34E − 05	7.25E − 28	1.06E − 18	0.000136	0.000202	2.53E − 16	9.67E − 17	1.15E − 38	0.2729 E − 37
F2	**6.97E − 169**	**0.00578**	7.18E + 17	0.029014	8.21E − 169	0.0077	0.042144	0.045421	0.055655	0.194074	0.1899 E − 22	0.2322 E − 22
F3	**3.03E − 304**	**0.000118**	3.29E + 06	79.14958	4.54E − 304	0.0098	70.12562	22.11924	896.5347	318.9559	0.1436 E − 6	0.3732 E − 6
F4	**1.89E − 157**	**3.02E − 157**	5.61E − 07	1.31508	5.89E − 157	7.02E − 157	1.086481	0.317039	7.35487	1.741452	0.5109 E − 9	0.8076 E − 9
F5	**21.0094**	0.089961	26.81258	69.9049	28.8342	**0.0811**	96.71832	60.11559	67.54309	62.22534	26.7856	1.8406
	Multimodal test functions											
F8	− **7.48E + 03**	− **6028.76**	− 6123.1	− 4087.44	− 2.36E + 03	4.11E + 02	− 4841.29	1152.814	− 2821.07	493.0375	− 5766.8	829.9
F9	**0**	**0**	0.31052	47.35612	**0**	**0**	46.70423	11.62938	25.96841	7.470068	0.0758 E − 13	0.1965 E − 13
F10	**2.13E − 15**	**5.59E − 16**	1.06E − 13	0.07783	4.32E − 15	6.49E − 16	0.276015	0.50901	0.062087	0.23628	0.1427 E − 13	0.0259 E − 13
F11	**0**	**0**	0.00448	0.006659	**0**	**0**	0.009215	0.007724	27.70154	5.040343	0.00331	0.000609

The bold values indicate the best results obtained for the average (Avg) and standard deviation (StD) calculated during the thirty runs. The values show the superiority of GGWO's results compared to other algorithms in almost all the functions used

evaluate the algorithms on the fuzzy control examples, as shown below.

5.2.1 Example 1: nonlinear plant-tracking control

In this example, the plant to be controlled is described by

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \tag{27}$$

where $-2 \le y(k) \le 2$, y (0) = 0, *u(k)* is the control input, and $-1 \le y(k) \le 1$. The main goal is to control the output *u(k)* to track the following desired trajectory:

$$y_d(k) = \sin\left(\frac{\pi k}{50}\right).\cos\left(\frac{\pi k}{50}\right) 1 \le k \le 250 \tag{28}$$

using a fuzzy controller system. In ANT [11], the ACO was used to select the consequent part of TSK-FSs from a predefined discrete set but could not track the trajectory in (26). The designed fuzzy controller inputs are $y_d(k+1)$ and $y(k)$. And the output is *u(k)*. For this tracking problem, RMSE is used for performance evaluation, where

$$\text{RMSE} = \sqrt{\frac{\sum_{k=0}^{249}(y_d(k+1) - y(k+1))^2}{250}} \tag{29}$$

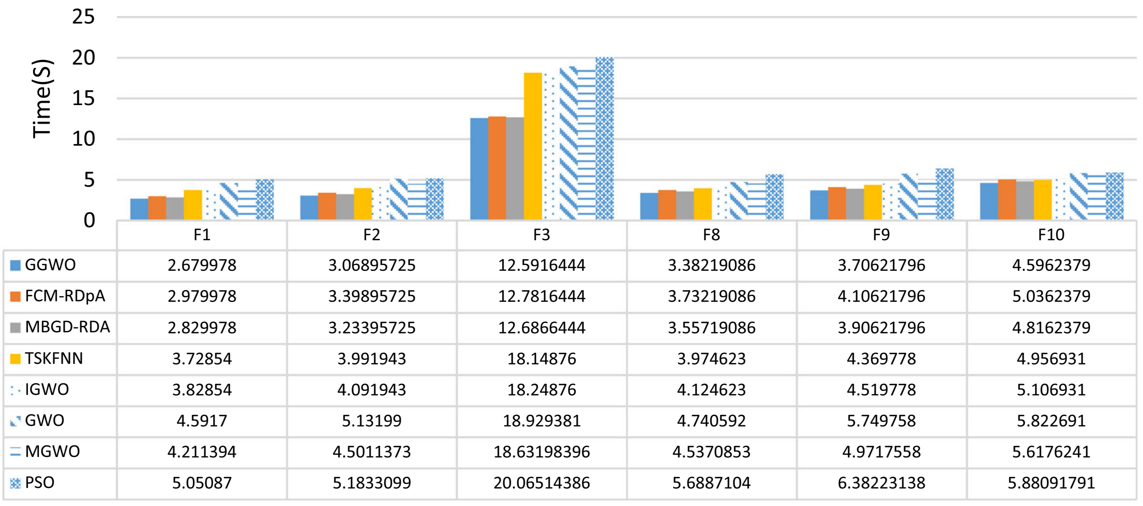
In this experiment, the optimization process is implemented through fifty runs and the maximum number of iterations is set to 500. The threshold Φ_{ThrS} is set to 0.0004. The average number of fuzzy rules (genes) is 6.

Figure 7 shows the RMSEs of the proposed GGWO algorithm compared to the PSO [8], GWO [13], IGWO [21], ANT [11], HCMSPSO [9], FCM-RDpA [37], MBGD-RDA [36], and TSKFNN [38] algorithms. To verify the performance of GGWO through comparisons with other algorithms, the number of rules and the threshold Φ_{ThrS} are set to the same values in all simulations, and the algorithms were limited to a maximum of 500 iterations. Figure 9 reveals the significant superiority of GGWO over the GWO-based, PSO, and ANT colony algorithms.

In addition, the tracking control results of GGWO and other algorithms using a first-order TSK-type fuzzy controller are compared and reported in Table 4.

Table 4 shows the averages (Avg) and standard deviations (StD) of the RMSEs over 50 runs. The GGWO achieves the smallest average RMSE. Moreover, Fig. 10 shows the RMSE values of each iteration obtained by different algorithms during the first-order TSK-type fuzzy system's optimization.

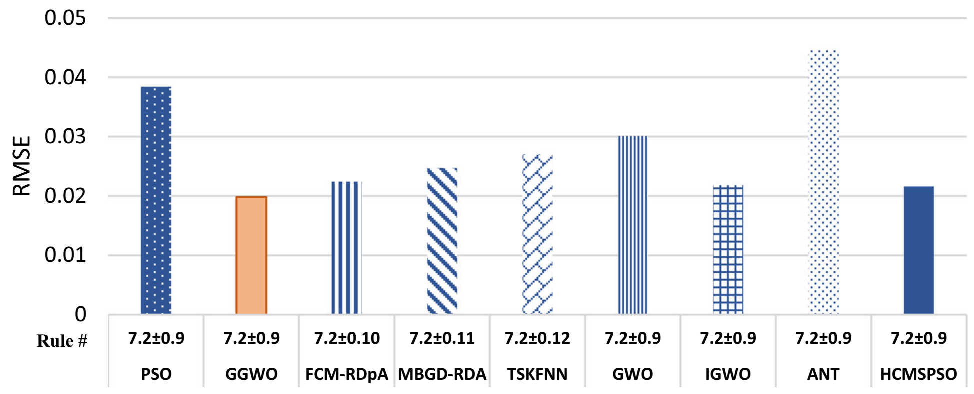
The obtained results demonstrate the increased capability of the proposed GGWO algorithm compared to the PSO [8], GWO [13], IGWO [21], ANT [11], and HCMSPSO [9] algorithms. These results occur because of



	F1	F2	F3	F8	F9	F10
GGWO	2.679978	3.06895725	12.5916444	3.38219086	3.70621796	4.5962379
FCM-RDpA	2.979978	3.39895725	12.7816444	3.73219086	4.10621796	5.0362379
MBGD-RDA	2.829978	3.23395725	12.6866444	3.55719086	3.90621796	4.8162379
TSKFNN	3.72854	3.991943	18.14876	3.974623	4.369778	4.956931
IGWO	3.82854	4.091943	18.24876	4.124623	4.519778	5.106931
GWO	4.5917	5.13199	18.929381	4.740592	5.749758	5.822691
MGWO	4.211394	4.5011373	18.63198396	4.5370853	4.9717558	5.6176241
PSO	5.05087	5.1833099	20.06514386	5.6887104	6.38223138	5.88091791

Fig. 8 Simulation Time of GGWO Compared to other algorithms

Fig. 9 Performance of GGWO compared to other GWO-based algorithms in Example 1



the previous modifications to the standard GWO algorithm. Figures 11 and 12 show the initial distributions of the fuzzy sets for the two input variables and the fuzzy sets' distributions after optimization by GGWO.

The * represents the fuzzy rules in the input space while the circles and ovals determine the distribution of the rules among the input space. Figure 11 shows the initial distributions of the fuzzy rules in the input space. Then, the final distributions of the rules in the input space are optimized after implementing the GGWO algorithm, as shown in Fig. 12.

Figure 13 shows the best fuzzy control results of the GGWO on Example 1, showing the desired and the actual tacking output using the GGWO algorithm.

5.2.2 Experiment three: testing the accuracy of the GGWO algorithm when applied to three real datasets

This experiment uses different datasets taken from the UC Irvine Machine Learning Repository, namely, Abalone [44], Airfoil [45], and PowerPlant [46]. Abalone dataset is used to predict the age of Abalone from physical measurements by cutting the shell through the cone, staining it,

Table 4 Tracking control (RMSE) by different ALGORITHMS ON Example 1

	GGWO	FCM-RDpA	MBGD-RDA	TSKFNN	HCMSPSO	IGWO	GWO	PSO	ANT
Avg.	0.033	0.039	0.036	0.04	0.04	0.044	0.041	0.045	0.059
StD.	0.012	0.012	0.011	0.008	0.008	0.0139	0.012	0.017	0.011

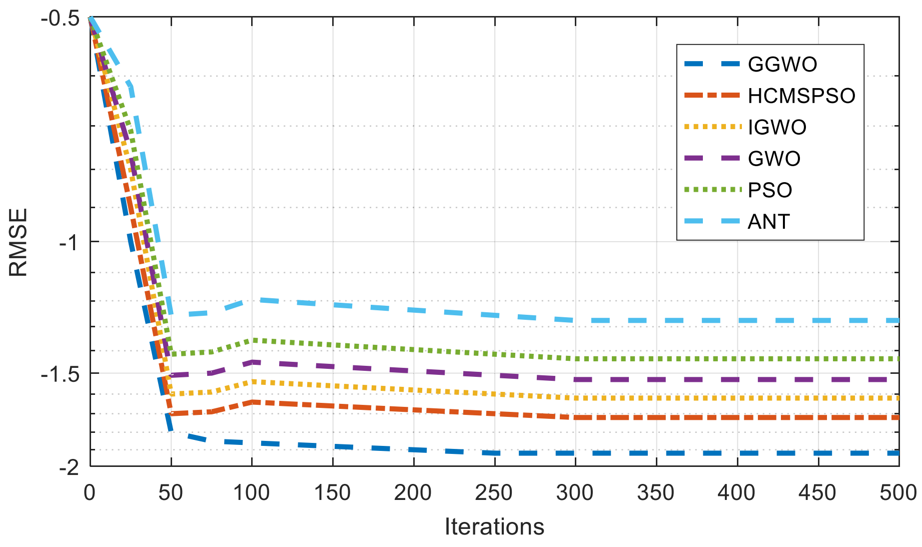


Fig. 10 Root-Mean-Squared-Error (RMSE) values on each iteration obtained by different algorithms in Example 1

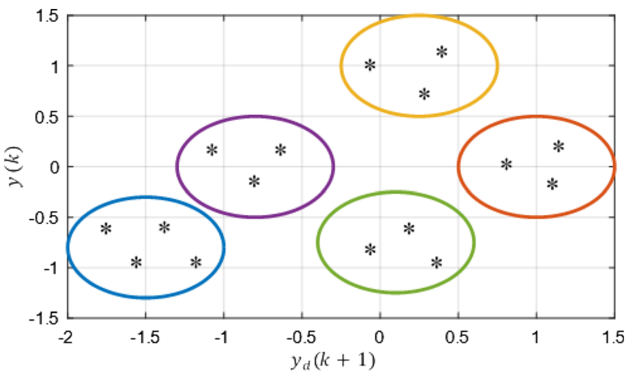


Fig. 11 Initial distributions of the rules in the input space by the rule generation method

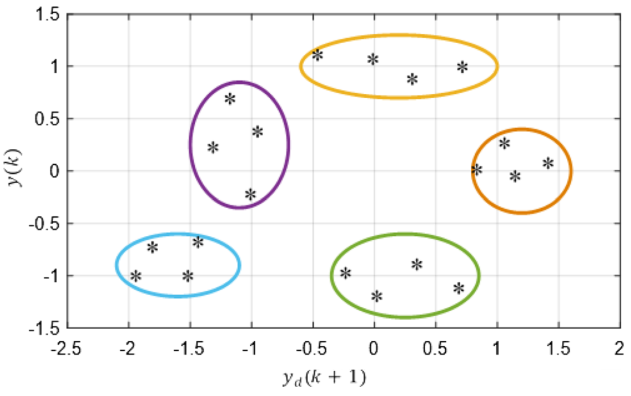


Fig. 12 The final distributions of the rules in the input space after optimization by GGWO

and counting the number of rings through a microscope. This task consumes time and effort, so other different measurements have been used to easily predict the age.

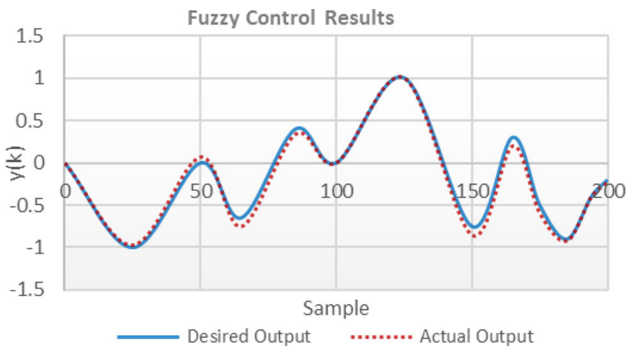


Fig. 13 Fuzzy control results of the GGWO in Example 1

Abalone dataset consists of eight input variables: sex, length, diameter, height, whole weight, shucked weight, viscera weight and shell weight; however, the first input variable is neglected because it is a trichotomous variable. The output variable is the number of rings that predict the age of Abalone. The dataset contains 4177 samples. Table 5 summarizes the properties of the three datasets used in the experiments.

This experiment's main goal is to test the accuracy of the GGWO algorithm when implemented on real datasets. The proposed algorithm is compared to different algorithms, namely FCM-RDpA [37], MBGD-RDA [36], TSKFNN [38], DE [47], Artificial Bee Colony (ABC) [48], which perform unconstrained optimization, as well as with the C-PSO [10], which performs constrained particle swarm optimization. The abalone dataset data are divided as follows: 70% are used as a training set and 30% as a test set.

Examples of the Abalone dataset's fuzzy partitions are illustrated in Fig. 14, where VW, W, M, S, and VS stand

Table 5 Description of the datasets used in the experiments

Dataset	Instances No.	Features No.	Features used No.	TSK parameters No.
Abalone	4177	8	5	212
Airfoil	1503	5	5	212
PowerPlant	9568	4	4	96

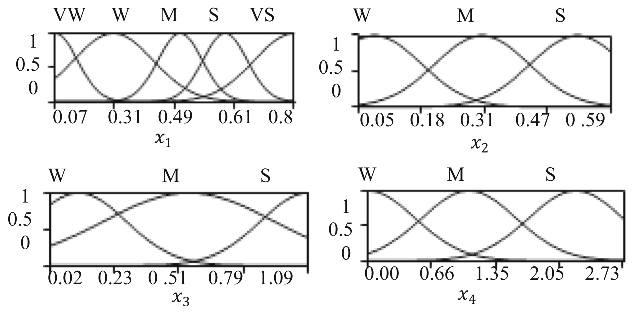


Fig. 14 Examples of fuzzy partitions of the dataset Abalone

for Very Weak, Weak, Medium, Strong, And Very Strong, respectively. The rule base equivalent to the fuzzy partitions of the abalone dataset in Fig. 12 is shown in Table 6.

The grid partitioning is implemented to divide the input space into a number of fuzzy partitions, each of which is specified by a membership function for each feature dimension. Figure 14 illustrates the input space fuzzy partitions for the Abalone dataset. In the implemented first-order TSK FSs, the Gaussian membership function represents the local information efficiently and contain continuous derivatives of the values. The consequent function $\mathbb{Z}_j(x)$ is set to a linear function of the input variables. The output of the system is given by Eq. (12).

Samples of the membership functions of the fuzzy partitions in the fuzzy rules generated by GGWO are shown in Fig. 15. The fuzzy rules can be described as shown in Table 6.

From Fig. 14, as the number of rules (c) increases, the probability of finding the best solution (F_{min}) decreases. To test the accuracy of GGWO, the average RMSE for training and testing data in abalone dataset is calculated and listed in Table 7, along with the number of rules for which it achieves the best accuracy and the time taken by the GGWO algorithm to compute all the attributes of the dataset.

In addition, the average RMSE for the GGWO algorithms and a different number of iterations on Abalone, Airfoil, and PowerPlant dataset are shown in Figs. 16, 17 and 18, respectively. The results obtained are compared to FCM-RDpA [37], MBGD-RDA [36], TSKFNN [38], GWO [13], IGWO [21], and HCMSPSO [9] algorithms.

The previous figures show that the GGWO algorithm performed the best among the other algorithms in the three datasets. The TSK-FS were trained well from the GGWO algorithm during the iteration. These results ensured that hybridizing the GA with GWO was useful in optimizing the learning rates, which in turn facilitated achieving enhanced learning performances.

To guarantee the fairness of the comparative results, the parameters of all algorithms are equalized as follows: The maximum number of iterations is 500, the population size is 30. The crossover and mutation rate is 0.7 and 0.01, respectively.

In addition, the computational cost in secs for the GGWO algorithm is calculated and compared with the algorithms that had the closest RMSE to our results, namely FCM-RDpA [37], MBGD-RDA [36], and TSKFNN [38] algorithm. Figure 19 shows the computational cost in seconds for the GGWO algorithm implemented into Abalone, Airfoil, and PowerPlant datasets.

The results show that the proposed algorithms were the faster among other algorithms in the three datasets.

Furthermore, to validate the obtained results, the GGWO results are compared with the results obtained by (DE) [47], (ABC) [48], and C-PSO [10], as shown in Fig. 20.

Figure 20 shows that the proposed GGWO algorithm results are comparable to those obtained by the specifically designed hybrid algorithms.

5.2.3 Experiment four: test the classification accuracy of the proposed GGWO-ANN algorithm

An Optimized Artificial Neural Network, proposed in [49], had been implemented along with GGWO algorithm to optimize the classification performance for the three datasets: Abalone, Airfoil, and PowerPlant. To validate the proposed GGWO-ANN algorithm, the results obtained are compared to MRC-TSK-FSC [35], FCM-RDpA [37], MBGD-RDA [36], SVM [13], and TSKFNN [38] algorithm. Table 8 summarizes the accuracy obtained, in terms of mean values and standard deviation, when implementing the ANN in different datasets.

The results show that the classification performance of GGWO-ANN is promising in the datasets with a relatively large number of instances. However, in the dataset with a few instances, as in Airfoil dataset, the accuracy is not the best among the other algorithms. The recent MRC-TSK-FSC [35] algorithm outperforms the GGWO-ANN

Table 6 Rule base equivalent to the fuzzy partitions of the abalone dataset in Fig. 12

*R*1	If × 1 is S and × 2 is M and × 3 is W and × 4 is W, Then y = 4.312 − 2.0108 × 1 + 5.409 × 2 + 18.988 × 3 + 20.0752 × 4
*R*2	If × 1 is VS and × 2 is W and × 3 is W and × 4 is M, Then y = 2.877 − 3.981 × 1 + 18.431 × 2 + 5.991 × 3 − 1.312 × 4
*R*3	If × 1 is VS and × 2 is W and × 3 is M and × 4 is S, Then y = 3.812 + 17.889 × 1 − 31.912 × 2 + 4.521 × 3 + 6.451 × 4
*R*4	If × 1 is W and × 2 is W and × 3 is M and × 4 is S, Then y = 1.561 − 16.609 × 1 + 10.319 × 2 + 2.951 × 3 + 1.981 × 4
*R*5	If × 1 is VW and × 2 is W and × 3 is S and × 4 is S, Then y = − 0.0897 − 0.0175 × 1 − 0.018 × 2 − 0.007 × 3 − 0.012 × 4
*R*6	If × 1 is M and × 2 is S and × 3 is W and × 4 is W, Then y = 6.471 + 17.592 × 1–14.401 × 2 + 9.951 × 3—2.519 × 4

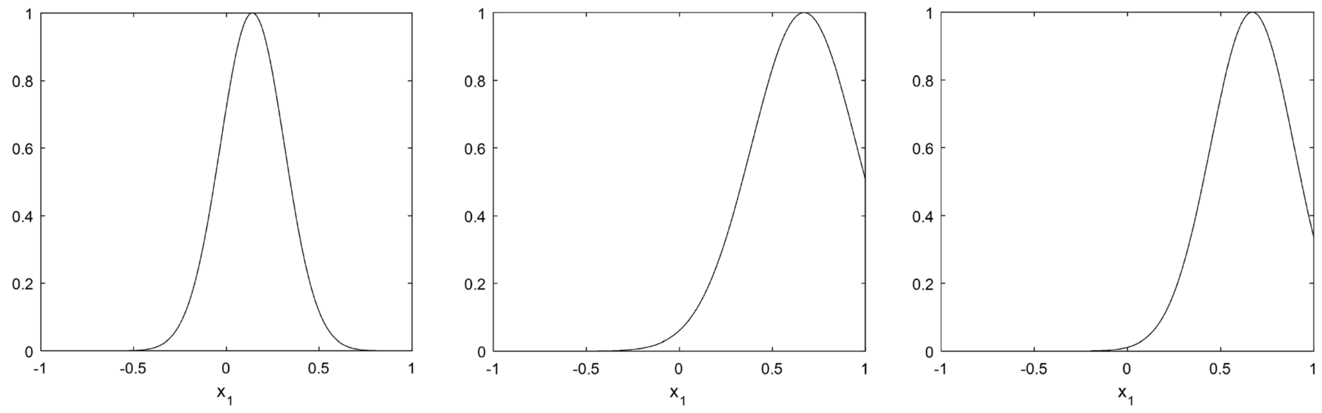


Fig. 15 Sample of membership functions of the fuzzy partitions in fuzzy rules generated by GGWO

Table 7 Optimal number of rules, computational cost, and RMSE values

Rules	Computational time (secs/iteration)	RMSE	
		Train	Test
6	1.198	1.9814	1.9967

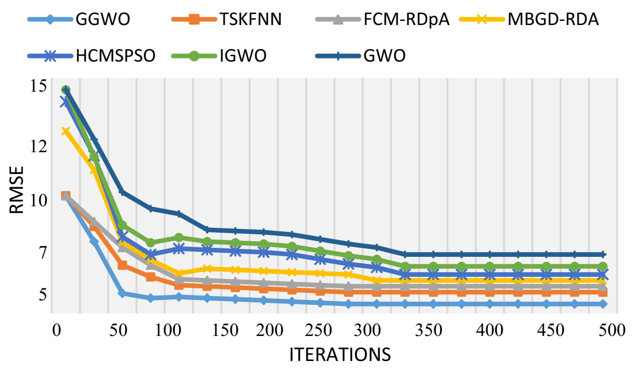


Fig. 16 Average RMSE for the GGWO algorithms along with different iterations on PowerPlant dataset compared to other algorithms

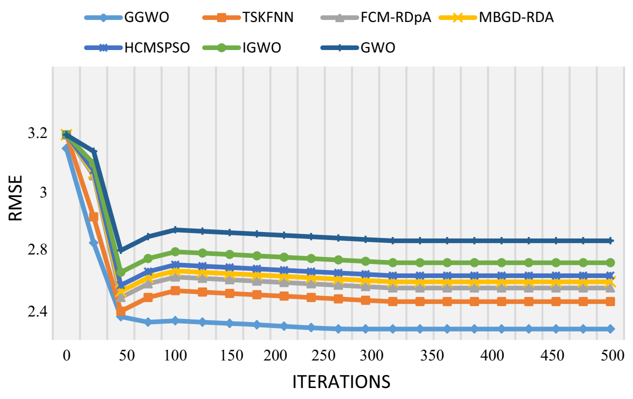


Fig. 17 Average RMSE for the GGWO algorithms along with different iterations on Abalone dataset compared to other algorithms

performance. A deep study for different datasets will be considered in future work.

5.2.4 Experiment Five: test the influence of the control parameters σ_{initW}, Φ_{ThrS}, and Θ on the performance of GGWO

During the TSK-FS optimization process, GGWO relies on some selected parameters: σ_{initW} is the initial width of the

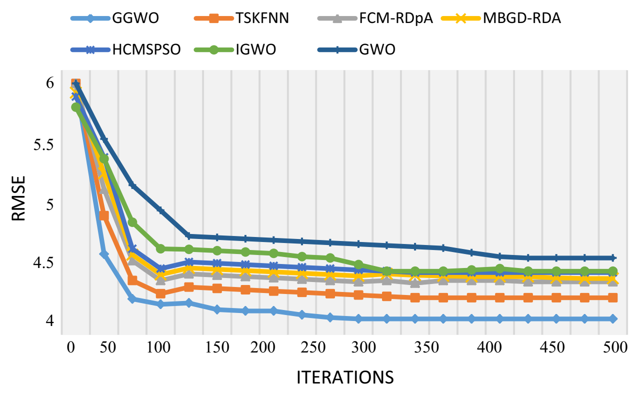


Fig. 18 Average RMSE for the GGWO algorithms along with different iterations on Airfoil dataset compared to other algorithms

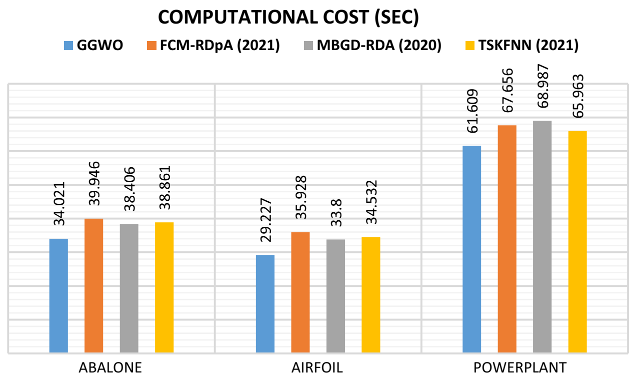


Fig. 19 Computational cost in seconds for the GGWO algorithm implemented into Abalone, Airfoil and PowerPlant datasets

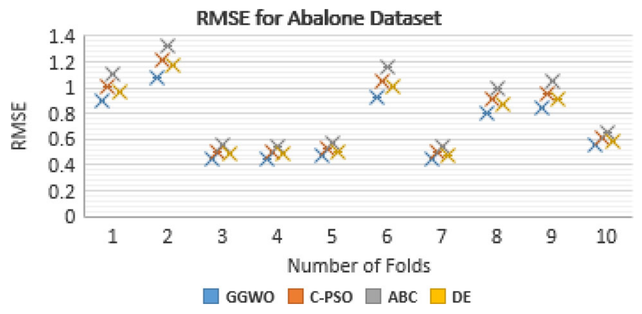


Fig. 20 RMSE for the GGWO algorithms within each fold on abalone compared to other algorithms

fuzzy set, Φ_{ThrS} is a predefined threshold for rule generation decision, and Θ is the degree of overlap between two rules. These three parameters (σ_{initW}, Φ_{ThrS}, and Θ) have large effects on the optimization performance because they influence the number of generated rules. In other words, the selection of the Φ_{ThrS}, σ_{initW}, Θ parameters directly affects the optimization performance of GGWO. In this experiment, the effects of changing the values of these parameters on the overall optimization performance of the proposed GGWO are investigated by calculating the resulting RMSE. The influence of Φ_{ThrS} and Θ is illustrated by examining the nonlinear plant-tracking control problem in Example 1. Figure 21 shows the GGWO performance for different values of Φ_{ThrS}, and the results are compared to HCMSPSO [9], C-PSO [10], MGWO [23], and IGWO [21].

Figures 22 and 23 show the RMSE values obtained by GGWO when optimizing the nonlinear plant-tracking control problem using different values of Θ and Φ_{ThrS}, respectively. The results are compared to HCMSPSO [9], C-PSO [10], MGWO [23], and IGWO [21].

As the figures show, the RMSE values remain stable as the values of Φ_{ThrS}, σ_{initW} and Θ increase until they reach a specific value. Subsequently, the RMSE values increase as the values of $\Phi_{ThrS}, \sigma_{initW,}$ and Θ increase. Consequently, this value can be considered as a threshold value for the GGWO. In addition, the previous figures show that GGWO achieves performance gains compared to the other algorithms.

To evaluate the performance of GGWO, other optimizing algorithms are also tested on the same dataset, examples and parameters. The experiments showed the RMSE, simulation time, computational cost, and the final distributions of the rules for different algorithms. The smaller RMSE, the less simulation time and the less computational cost indicate that the algorithm has better performance. It can be seen from the experiments that GGWO obtains the smallest RMSE in Abalone, PowerPlant, and Airfoil datasets and has a better performance compared to others in running the functions, with less timing compared to others.

6 Conclusion

This paper proposed a hybrid GGWO algorithm for first-order TSK-FS optimization. GGWO combines the crossover and mutation operations of GA with the exploration and exploitation stages of GWO. This combination has been shown to have several advantages: (1) partitioning the solution into several genes (chromosomes) improved the diversity search of GGWO; (2) the combination maintains a balance between solution exploitation and exploration; (3) the exploration process is accelerated and reaches the best solution in less time; (4) the GGWO algorithm overcomes the tendency of premature convergence and the poor exploitation of solutions in the standard GWO algorithm.

Table 8 Classification accuracy of GGWO-ANN compared to different algorithms

Dataset		GGWO-ANN	MRC-TSK-FSC	FCM-RDpA	MBGD-RDA	TSKFNN	SVM
Abalone	Mean	**0.9838**	0.9796	0.9701	0.9781	0.967	0.9676
	Std.	– 0.0145	– 0.0149	– 0.0141	– 0.012	– 0.0139	– 0.0145
Airfoil	Mean	0.7709	**0.7909**	0.7731	0.7608	0.7644	0.76
	Std.	– 0.02985	– 0.0287	– 0.031	– 0.0823	– 0.0419	– 0.0391
PowerPlant	Mean	**0.9631**	0.9132	0.8812	0.8986	0.899	0.8909
	Std.	– 0.0161	– 0.0191	– 0.0131	– 0.0091	– 0.0312	– 0.016

The bold values indicate the best results obtained for the mean values calculated for testing the classification accuracy of the algorithms. The values show the superiority of GGWO's results in Abalone and PowerPlant dataset compared to other algorithms, while MRC-TSK-FSC algorithm outperform the GGWO-ANN performance in Airfoil dataset

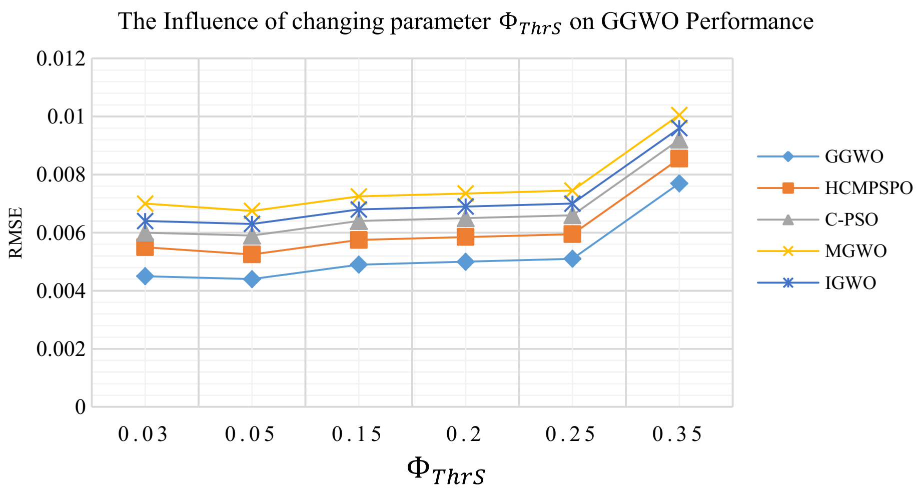


Fig. 21 Influence of changing parameter Φ_{ThrS} on GGWO Performance

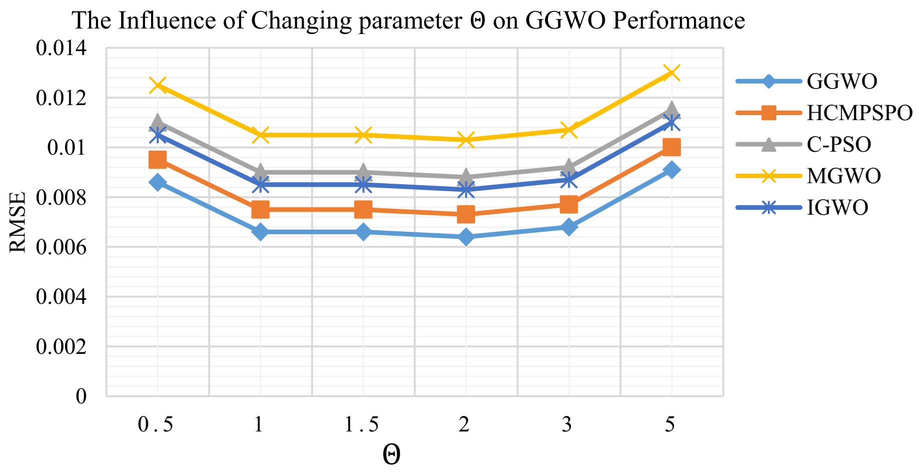


Fig. 22 Influence of changing parameter Θ on GGWO Performance

Unimodal and multimodal benchmark functions are used to demonstrate the GGWO's efficiency. The results validate that GGWO achieves better values than other nature-inspired optimization algorithms. In addition, GGWO is applied to generating fuzzy rules for TSK-FSs and optimizing the parameters for each fuzzy rule. GGWO tunes all the free parameters in the designed TSK-FSs. The simulation results show that the GGWO can optimally design TSK-FSs and achieve higher accuracy than other optimization algorithms. We intend to apply GGWO algorithm to different multi-objective optimization, multi-constraint optimization, and dynamic uncertainty.

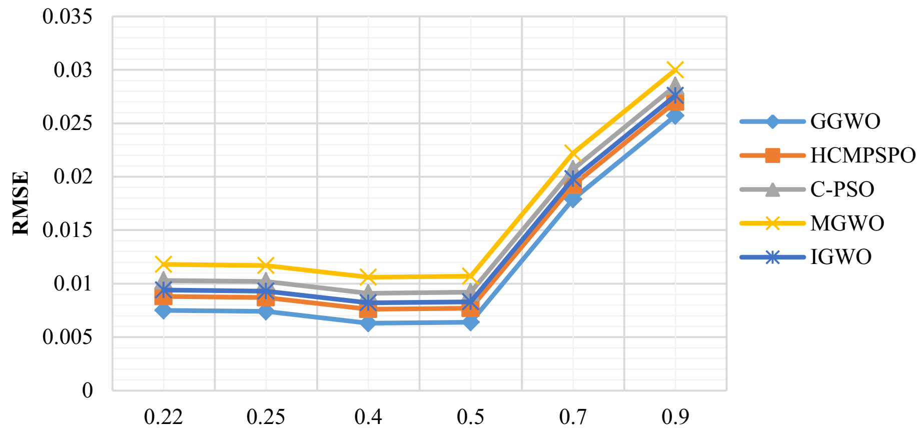


Fig. 23 Influence of changing parameter σ_{initW} on GGWO Performance

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Zadeh LA (1965) Fuzzy sets. Inf Control 8(3):338–353. https://doi.org/10.1016/S0019-9958(65)90241-X
2. Juang CF, Chiu SH, Chang SW (2007) A self-organizing TS-type fuzzy network with support vector learning and its application to classification problems. IEEE Trans Fuzzy Syst 15(5):998–1008
3. Yao X (ed) (1999) Evolutionary computation: theory and applications. World Scientific, Singapore
4. Bäck T (1996) Evolutionary algorithms in theory and practice. Oxford University Press, New York
5. Juang CF (2002) A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. IEEE Trans Fuzzy Syst 10(2):155–170
6. Hoffmann F, Schauten D, Holemann S (2007) Incremental evolutionary design of TSK fuzzy controllers. IEEE Trans Fuzzy Syst 15(4):563–577
7. Mansoori EG, Zolghadri MJ, Katebi SD (2008) SGERD: A steadystate genetic algorithm for extracting fuzzy classification rules from data. IEEE Trans Fuzzy Syst 16(4):1061–1071
8. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. 1995, pp. 1942–1948
9. Juang CF, Hsiao CM, Hsu CH (2010) Hierarchical cluster-based multispecies particle-swarm optimization for fuzzy-system optimization. IEEE Trans Fuzzy Syst 1(18):14–26
10. Tsekouras GE, Tsimikas J, Kalloniatis C, Gritzalis S (2017) Interpretability constraints for fuzzy modeling implemented by constrained particle swarm optimization. IEEE Trans Fuzzy Syst 26(4):2348–2361
11. Juang CF, Lu CM, Lo C, Wang CY (2008) Ant colony optimization algorithm for fuzzy controller design and its FPGA implementation. IEEE Trans Ind Electron 55(3):1453–1462
12. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179:2232–2248
13. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
14. Precup R, David R, Petriu EM (2017) An easily understandable grey wolf optimizer and its application to fuzzy controller tuning. Algorithms 10(2):68. https://doi.org/10.3390/a10020068
15. Rodr L, Castillo O, Soria J (2017) Optimizer, GW: a study of parameters of the grey wolf optimizer algorithm for dynamic adaptation with fuzzy logic. Springer, Berlin
16. Noshadi J, Shi WS, Lee PS, Kalam A (2015) Optimal PID-type fuzzy logic controller for a multi-input multi-output active magnetic bearing system. Neural Comput Appl. https://doi.org/10.1007/s00521-015
17. Mirjalili S (2015) How effective is the grey wolf optimizer in training multi- layer perceptrons. Appl Intell 43(1):150–161
18. Saremi S, Mirjalili SZ, Mirjalili SM (2015) Evolutionary population dynamics and grey wolf optimizer. Neural Comput Appl 26(5):1257–1263
19. Gupta S, Deep K (2019) Hybrid grey wolf optimizer with mutation operator. In: Bansal J, Das K, Nagar A, Deep K, Ojha A (eds) Soft computing for problem solving. Advances in intelligent systems and computing. Springer, Singapore
20. Holland JH (1975) Adaptation in natural and artificial systems: an introductory analysis with application to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor, MI
21. Sahoo BP, Panda S (2018) Improved grey wolf optimization technique for fuzzy aided PID controller design for power system frequency control, Sustainable Energy. Grids Networks. https://doi.org/10.1016/j.segan.2018.09.006

22. Tawhid MA, Ali AF (2017) Regular research paper, A Hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function. Memetic Comput 9:347–359. https://doi.org/10.1007/s12293-017-0234-5
23. Padhy S, Panda S, Mahapatra S (2017) A modified GWO technique based cascade PI-PD controller for AGC of power systems in presence of Plug in Electric Vehicles. Int J Eng Sci Technol 20:427–442
24. Elghamrawy SM, Hassanien AE (2017) A partitioning framework for Cassandra NoSQL database using Rendezvous hashing. J Supercomput 73(10):4444–4465
25. Zhou SM, Gan JQ (2008) Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modeling. Fuzzy Sets Syst 159(23):3091–3131
26. Cordón O, Gomide F, Herrera F, Hoffmann F, Magdalena L (2004) Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy Sets Syst 141(1):5–31
27. Chen T, Shang C, Su P, Shen Q (2018) Induction of accurate and interpretable fuzzy rules from preliminary crisp representation. Knowl-Based Syst 146:152–166
28. Hein D, Udluft S, Runkler TA (2018) Generating interpretable fuzzy controllers using particle swarm optimization and genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 1268–1275). ACM
29. Wiktorowicz K, Krzeszowski T, Przednowek K (2021) Sparse regressions and particle swarm optimization in training high-order Takagi-Sugeno fuzzy systems. Neural Comput Appl 33(7):2705–2717
30. Shao Y, Lin JCW, Srivastava G, Guo D, Zhang H, Yi H, Jolfaei A (2021) Multi-objective neural evolutionary algorithm for combinatorial optimization problems. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2021.3105937
31. David RC, Precup RE, Preitl S, Szedlak-Stinean AI, Roman RC, Petriu EM (2020) Design of low-cost fuzzy controllers with reduced parametric sensitivity based on whale optimization algorithm. 2020 IEEE international conference on fuzzy systems (FUZZ-IEEE). IEEE, pp 1–6
32. Lin JCW, Hong TP, Lin TC (2015) A CMFFP-tree algorithm to mine complete multiple fuzzy frequent itemsets. Appl Soft Comput 28:431–439
33. Wiktorowicz K, Krzeszowski T (2022) Identification of time series models using sparse Takagi-Sugeno fuzzy systems with reduced structure. Neural Comput Appl 34:7473–7488. https://doi.org/10.1007/s00521-021-06843-5
34. Deng Z, Cao Y, Lou Q, Choi KS, Wang S (2022) Monotonic relation-constrained Takagi-Sugeno-Kang fuzzy system. Inf Sci 582:243–257
35. Hušek P (2022) Monotonic Takagi-Sugeno models with cubic spline membership functions. Expert Syst Appl 188:115997
36. Wu D, Yuan Y, Huang J, Tan Y (2020) Optimize TSK fuzzy systems for regression problems: Minibatch gradient descent with regularization, DropRule, and AdaBound (MBGD-RDA). IEEE Trans Fuzzy Syst 28(5):1003–1015
37. Shi Z, Wu D, Guo C, Zhao C, Cui Y, Wang FY (2021) FCM-RDpA: tsk fuzzy regression model construction using fuzzy c-means clustering, regularization, droprule, and powerball adabelief. Inf Sci 574:490–504
38. Wang T, Gault R, Greer D (2021) A novel Data-driven fuzzy aggregation method for Takagi-Sugeno-Kang fuzzy Neural network system using ensemble learning. In: 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (pp. 1–6). IEEE
39. Du A, Shi X, Guo X, Pei Q, Ding Y, Zhou W, Lu Q, Shi H (2021) Assessing the adequacy of hemodialysis patients via the graph-based Takagi-Sugeno-Kang fuzzy system. Comput Math Methods Med 2021:1–17
40. Muro C, Escobedo R, Spector L, Coppinger R (2011) Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations. Behav Process 88:192–197
41. Lin CT, Lee CSG (1996) Neural fuzzy systems: a neural-fuzzy synergism to intelligent systems. Prentice-Hall, Englewood Cliffs
42. Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor
43. Colin RR, Jonathan ER (2002) Genetic algorithms-Principles and perspectives, A guide to GA Theory. Kluwer Academic Publisher, Amsterdam
44. http://archive.ics.uci.edu/ml/datasets/Abalone
45. https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise
46. https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant
47. Storn R, Price K (1997) Differential evolution- a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
48. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function approximation: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459–471
49. Engy EL, Ali EL, Sally EG (2018) An optimized artificial neural network approach based on sperm whale optimization algorithm for predicting fertility quality. Stud Inf Control 27(3):349–358