International Journal of Mathematical, Engineering and Management Sciences
Vol. 11, No. 4, 1680-1717, 2026
https://doi.org/10.33889/IJMEMS.2026.11.4.069

Modified Yellow Saddle Goatfish Algorithm for Problem Solving in Optimization

Taruna Chugh
Department of Mathematics,
Chandigarh University, Gharuan, 140413, Mohali, Punjab, India.
E-mail: taruna.e1607@cumail.in

Ashok Pal
Department of Mathematics,
Chandigarh University, Gharuan, 140413, Mohali, Punjab, India.
E-mail: ashoke4593@cumail.in

Sumita Gulati
Department of Mathematics,
S.A. Jain (P.G.) College, 134001, Ambala City, Haryana, India.
Corresponding author: sumitabansal@sajaincollege.ac.in

(Received on December 24, 2025; Revised on March 10, 2026 & May 17, 2026 & June 29, 2026; Accepted on July 2, 2026)

Abstract
Yellow Saddle Goatfish Algorithm (YSGA) is a new metaheuristic algorithm, which attempts to model the cooperative hunting behaviour of yellow saddle goatfish to solve optimization problems. It has been found to be very effective in exploring complicated search spaces. An improved version of YSGA is offered in this work where the zone changing stage which is one of the most important components which assure the determination of the balance between the exploration and the exploitation is enhanced in a systematic manner to enhance the convergence behaviour. The principal goal behind changing the zone-changing mechanism will be to move faster towards the global optimum and minimize the likelihood of premature convergence towards local optima. The modified algorithm is highly tested across 23 standard benchmark functions of different dimension coded in MATLAB. The convergence rate, local minima, and trade-off between exploration and exploitation are examined with regard to performance. Comparative results indicate that the modified YSGA improves upon the original YSGA and demonstrates competitive performance against well-known algorithms such as Cuckoo Search, Grey Wolf Optimization and Moth-Flame Optimization in terms of fitness values and convergence behaviour. Moreover, the applicability of the modified approach is proven by the fact that it is applied to three engineering optimization problems. Besides, statistical test is performed to ensure validity of the findings as well. Altogether, the noticed improvements prove the high capability of the modified YSGA to solve the large-scale optimization issues and give a promising background to the further evolution of the metaheuristic optimization techniques.

Keywords- Optimization algorithms, Zone changing phase, Yellow saddle goatfish, Exploration, Exploitation.

1. Introduction

Optimization is referred as the procedure of choosing most effective solution from given feasible possibilities for a given issue. While dealing with any optimization issue, three most important factors to be considered are fitness function, constraints and decision variables (Leung et al., 2023). In such cases, aim is to optimize fitness function by quantifying the decision variables in accordance to given constraints (Dehghani et al., 2022). Nevertheless, the importance and functionality of optimization in various scientific fields have become more evident with advancements in technology and research. As a result, efficient techniques are needed for tackling different optimization issues. Recently, academics have proposed various methods for solving optimization challenges. Certain optimization issues are quite

complicated and call for an ideal solution to be found in an appropriate period of time. In cases where an optimization problem has multiple local optima, the solution could end up in a local optimum, and the result could be contingent upon the initial fitness (Mohammadi & Sheikholeslam, 2023). Furthermore, the gradient search could grow unsteady in instances where the objective function has several distinct peaks. Considering this, investigators are exploring meta-heuristic optimization approaches because of their growing acceptance as a result of the shortcomings encountered with conventional methods, particularly for complex optimization issues (Salgotra et al., 2024). Meta-heuristic algorithm, which is sometimes also called as stochastic techniques are becoming increasingly popular and have been utilized in optimization problems because of their simple implementation, problem independence, efficacy for non-linear, non-convex systems and non-linear searching space (Sergeyev et al., 2018). Meta-heuristic is not confined to one type of optimization technique and hence is further divided into four categories of evolution, swarm, physics and human based techniques. Evolutionary algorithms are based on evolution techniques. Some of the prime examples of this category are Genetic Algorithm (GA) (Holland, 1992). Biogeography based optimizer (BBO) (Simon, 2008) and Differential Evolution (DE) (Storn & Price, 1997). Moreover, swarm algorithms mimic the social behavior of living creatures for solving optimization issues like Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995), Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016), Grasshopper Optimization Algorithm (GOA) (Saremi et al., 2017), Krill Herd (Gandomi & Alavi, 2012), Simplified Dolphin Echolocation (SDE) (Kaveh & Hosseini, 2014) and Grey Wolf Optimization (GWO) (Mirjalili et al., 2014a). On the other hand, physics and human based algorithms replicate physical and human behaviors respectively for solving optimization problems. Some examples of physical approaches are Big Bang-Big Crunch (BB-BC) (Erol & Eksin, 2006), Colliding Bodies Optimization (CBO) (Kaveh & Mahdavi, 2014), Gravitational Search Algorithm (GSA) (Rashedi et al., 2009), Star Graph (Gharebaghi et al., 2017), Water Wave Optimization (WWO) (Zheng, 2015) and Ray Optimization (Kaveh & Khayatazad, 2012), while as, Tabu Search (TS) (Glover, 1989), Human Mental Search (HMS) (Mousavirad & Ebrahimpour-Komleh, 2017) and Teaching Learning Based Optimization (TLBO) (Rao et al., 2011) are examples of Human based meta-heuristic algorithms.

Moreover, there are other algorithms such as Cuckoo Search (CS), Moth Flame Optimization (MFO) and Yellow Saddle Goatfish Algorithm (YSGA) that have drawn attention among the researchers during the last few years, due to their distinctive method of hunting. For instance, CS mimics the behavior of cuckoo bird and utilizes levy flights which is a random walk process with step sizes that follow a heavy-tailed probability distribution, which enhances the exploration capabilities of the search process (Joshi et al., 2017). This strategy allows CS to efficiently explore the search space and keep away from local minima, making it particularly robust for solving nonlinear, multimodal, and noisy optimization problems.

The other nature-inspired algorithm is MFO that is inspired by the behavior of moths in their movement around a flame i.e. the moths move in a logarithmic spiral manner (Mirjalili, 2015). The algorithm is flexible to various problems and has a fast speed of convergence in lower-dimensional space and can attain solutions that are close to the global optimum. The performance of MFO however can be extremely reliant on the dimensionality of the problem. In higher dimensions, the spiral motion that is central to the success of MFO can no longer be useful, and exploration-exploitation tradeoffs can arise.

Lastly, YSGA is another new optimization algorithm that is based on the pack hunting of the yellow saddle Goatfish (Zaldivar et al., 2018). YSGA is developed to move through intricate search spaces by imitating the hunt coordination of goatfish and, as a result, it can dynamically adjust its search pattern to locate optimal solutions. Nevertheless, the primary issues related to YSGA are its comparatively slow convergence rate where the algorithm might not be able to reduce quickly towards the global optimum.

The zone-changing step is a significant part of YSGA, governing the process by which the search process changes its focus to other parts of the solution space. But, as it was initially designed, this mechanism might not adapt well in complex search situations. To resolve this issue, the present work introduces a better zone-changing strategy that incorporates a memory-based update and controlled stochastic variations. Also, a condition-based mechanism is used to trigger zone transitions only when no more improvement can be noticed. These changes make the search process more flexible and help to maintain a balance between exploration and exploitation.

Despite the natural inspiration of all optimization algorithms, they are all based on two concepts: intensification and diversification. Diversification searches the space at random, in an attempt to find global optimum and the process of intensification is aimed at narrowing the search to the best solution at hand. Particularly, the consideration of the randomness of the nature of optimization processes makes it especially difficult to balance these two aspects.

Contribution and novelty of the work are

- A modified zone-changing process is developed by using information on past location to create a more stable search process.
- Stochastic perturbation mechanism is proposed to enhance exploration and keep solution diversity.
- A condition based switching strategy is structured in such a way that the zone transitions are activated only when there is no improvement.
- The performance of the modified method is assessed on various benchmark problems and compared against well-established optimization algorithms.

The remaining paper is organized as follows. Section 2 reviews the related literature, and Section 3 introduces the relevant terminologies. The proposed methodology is presented in Section 4. Section 5 reports the experimental setup, results, and statistical analysis, while Section 6 discusses constraint handling and applications. Finally, Section 7 concludes the paper and outlines future research directions.

2. Literature Review

In recent years, metaheuristic optimization algorithms have received a tremendous interest because of their ability to solve complex engineering and real-world optimization problems effectively. The high level of usage is due to their flexibility, simplicity and ability to give close-to-optimal solutions in a nonlinear and high dimensional search space. Despite their merits, traditional metaheuristic approaches have some drawbacks, including premature convergence, lack of diversity, and an imbalance between exploration and exploitation. To overcome these problems, various improvement approaches have been suggested such as chaotic mechanism, adaptive learning mechanism, hybridization method, opposition learning, and dynamic control. Recently, studies have primarily focused on enhancing convergence speed, global search ability and solution accuracy while maintaining computational efficiency in this context.

In this context, there has been in-depth research on optimizing the Yellow Saddle Goatfish Algorithm (YSGA) and its variants. Kashyap et al. (2021) added chaotic approach to improve the global optimization ability of YSGA by adding chaotic maps in the search process. These changes helped its exploration ability and enhance the diversity of the population, which reduced the risk of premature convergence. In a similar way, Leifu & Xuejiao (2021) introduced an enhanced YSGA that combines a decreasing strategy and chaotic mechanisms to control the intensity of search on iterations. The hybrid method increased stability in convergence and increased the optimization performance overall. Further, Utama et al. (2022) proposed an extended YSGA that can be applied to the vehicle routing problem (VRP) incorporating simultaneous pickup and delivery and fuel consumption constraints, which resulted

in a better routing efficiency. YSGA has been applied in another application domain by Brahma et al. (2022) in spectrum allocation for cognitive radio networks, where they achieved better spectrum utilization and decreased interference. All of these studies show the versatility of YSGA in both algorithmic improvements and practical optimization problems.

In addition to the YSGA improvements, researchers have also investigated general learning based mechanisms for improving the performance of metaheuristic. To enhance the convergence behavior, Jia & Lu (2024) have adopted a novel guided learning strategy as a new update mechanism to guide candidate solutions to the promising regions in the search space. Nayak et al. (2024) presented an adaptive automatic cuckoo search algorithm, which adaptively adjusts search parameters to adjust it in a good manner to balance the exploration and exploitation. This adaptive control was very helpful in improving robustness and avoiding premature convergence. These methods collectively support the need of smart learning mechanisms to enhance the efficiency of optimization.

At the same time a number of studies have been directed towards the hybridization of algorithms and towards chaos-based approaches to further enhance the optimization performance. Varshney et al. (2024) proposed dynamic random walk and opposition-based learning in Aquila Optimizer to boost the diversity and convergence character of population in constrained engineering issues. To enhance the search stability, Hashim et al. (2024) suggested a modified Sea Horse Optimizer with adaptive exploration and exploitation strategies. Kusuma & Purboyo (2025) put forward an Adaptive Iteration Algorithm and showed how it can be applied to the Economic Emission Dispatch problem, where the iteration mechanism is changed adaptively to achieve better convergence speed and solution quality. To enhance the adaptability of the algorithm, Zhang & Cai (2024) proposed Adaptive Dynamic Self-Learning Grey Wolf Optimization Algorithm (ADSL-GWO) that combines the nonlinear convergence control, spiral search and self-learning mechanisms. The technique improved the diversity of population and guaranteed exploration/exploitation balance.

Other significant improvements in hybrid metaheuristics have been widely studied too. Lu et al. (2020) introduced a chaotic Grey Wolf Optimizer, which makes use of chaotic maps to boost the randomness and exploration capability. To enhance the convergence accuracy, Tu et al. (2023) proposed a hybrid of the Grey Wolf and Harris Hawks Optimizers, which is a combination of the exploration and exploitation abilities of both algorithms. Ding et al. (2019) have added PSO and Cuckoo Search to improve the convergence rate and robustness. Likewise, Naik et al. (2016) proposed a CS–GSA hybrid algorithm and Tawhid & Ali (2018) introduced a Firefly–Cuckoo Search hybrid algorithm with better global search performance results. Gharehchopogh et al. (2023) proposed a chaotic quasi-oppositional farmland fertility algorithm, which further demonstrated that chaos and opposition based learning is effective to avoid local optima. Overall, all hybrid strategies show that the optimization capability is greatly improved in complex search spaces when the algorithmic fusion methods are used.

There are also a number of studies that have investigated hybrid optimization in engineering and machine learning applications that have been done alongside the algorithmic development. To improve engineering design accuracy and learning efficiency, Zhang et al. (2020) have suggested a hybrid teaching–learning-based optimization method for engineering design based on neural network. To improve the quality of obtained solutions, Mirjalili et al. (2014b) proposed a binary optimization algorithm based on a hybrid of Particle Swarm Optimization (PSO) and Gravitational Search Algorithm (GSA). An evolutionary optimization-based PWM method for hybrid power converter was proposed by Rodríguez et al. (2020) to improve system efficiency. From the results of these studies, it can be

concluded that hybrid metaheuristic algorithms have great applicability in solving practical engineering problems.

Table 1 summarizes the recent optimization algorithms, their key modifications, and the corresponding performance improvements discussed in the literature.

Table 1. Performance assessment of different optimization algorithms.

Reference	Optimization algorithm	Key modifications	Remarks
Kashyap et al. (2021)	YSGA	Chaotic maps integration	Improved convergence rate on benchmark functions
Leifu & Xuejiao (2021)	YSGA	Declining strategy and chaotic mechanism	Enhanced search efficiency and convergence behaviour
Utama et al. (2022)	Hybrid YSGA	Applied to vehicle routing problem with pickup-delivery and fuel constraints	Improved routing efficiency and reduced operational cost
Brahma et al. (2022)	YSGA	Applied to spectrum allocation in cognitive radio networks	Improved spectrum utilization and reduced interference
Jia & Lu (2024)	Guided Learning Strategy (GLS)	Novel guided updated mechanism for metaheurestic search direction	Improve convergence performance and solution guidance
Nayak et al. (2024)	Adaptive Cuckoo Search	Automatic adaptive parameter tuning for exploration-exploitation balance	Improved robustness and reduced premature convergence
Varshney et al. (2024)	Aquila Optimizer	Dynamic random walk with adaptive opposition-based learning	Improved balance between exploration and exploitation in constrained problems
Hashim et al. (2024)	Modified Sea Horse Optimizer	Adaptive exploration and exploitation control strategy	Improved convergence speed and solution quality
Kusuma & Purboyo (2025)	Adaptive Iteration Algorithm	Iteration-level adaptive parameter control	Improved convergence speed and computational efficiency
Zhang & Cai (2024)	Adaptive Dynamic Self-Learning GWO	Nonlinear convergence control and self-learning strategy	Improved population diversity and search balance
Lu et al. (2020)	Chaotic GWO	Integration of chaotic maps in GWO	Improved randomness and global search capability
Tu et al. (2023)	GWO-HHO Hybrid	Hybridization of grey wolf and Harris hawks optimizers	Improved convergence accuracy and exploitation strength
Ding et al. (2019)	PSO–CS Hybrid	Combination of particle swarm optimization and cuckoo search	Improved convergence speed and stability
Naik et al. (2016)	CS-GSA Hybrid	Hybrid of cuckoo search and gravitational search algorithm	Improved global exploration ability
Tawhid & Ali (2018)	Firefly–Cuckoo Search Hybrid	Hybrid swarm intelligence approach	Improved global search and solution quality
Gharehchopogh et al. (2023)	CQFFA	Chaotic quasi-oppositional farmland fertility algorithm	Improved ability to escape local optima
Zhang et al. (2020)	TLBO–Neural Network	Hybrid learning-based optimization for engineering design	Improved prediction and optimization accuracy
Mirjalili et al. (2014b)	PSO–GSA Hybrid	Binary hybrid optimization approach	Improved global search capability in discrete space
Rodríguez et al. (2020)	Evolutionary PWM Optimization	Evolutionary algorithm-based control strategy	Improved efficiency of hybrid power converter systems
Zamani et al. (2024)	Moth-Flame Optimization (MFO) Review	Comprehensive review of variants and performance analysis	Provides benchmark comparison and improvement directions
Sharma et al. (2023)	Butterfly Optimization Algorithm	Multi-objective non-dominated sorting strategy	Improved pareto solution diversity
Hosseinalipour et al. (2024)	CHIO Algorithm	Coronavirus herd immunity-based optimization model	High accuracy in medical classification tasks
Goel et al. (2023)	Image Fusion Optimization Survey	Optimization-based image fusion techniques review	Improved image quality and fusion performance understanding

Furthermore, a number of detailed reviews and special area optimization studies have been added to the literature. Zamani et al. (2024) gave a critical review of moth-flame optimization algorithm, which involved analysis of variants, performance behavior and improvement strategies. For multi-objective problems, Sharma et al. (2023) introduced a non-dominated sorting butterfly optimization algorithm,

which enhances Pareto optimality and diversity of solutions. In the area of breast cancer diagnosis, Hosseinalipour et al. (2024) proposed a Coronavirus Herd Immunity Optimizer (CHIO) based method which showed high performance in medical classification tasks. Goel et al. (2023) provided a survey of optimization-based image fusion techniques, emphasizing on the advantages of the image fusion techniques to enhance the quality of the image and the accuracy of the image fusion. Even with the tremendous progress made in optimization by means of metaheuristics, there are still some research gaps, during the development of the Yellow Saddle Goatfish Algorithm. While hybridization can be explored to increase the exploration capability, it can also add complexity to the structure, thus increasing the computational cost, and reducing the algorithm efficiency.

Moreover, despite the pack-hunting behaviour and optimization potential that has garnered much attention for YSGA, it has some drawbacks, including slow convergence and the potential for getting stuck in local optima. Past enhancement methods like chaotic maps, adaptive control methods and hybrid methods have led to a certain level of performance but have failed to provide an easy-to-implement yet efficient way for fast convergence and strong exploration. Apart from that, some modifications like chaotic initialization, adaptive parameter tuning and hybrid integration have been proposed but the zone changing phase, which is important for balancing exploration and exploitation, is not sufficiently explored.

Hence, the need for a more simplified, yet efficient YSGA variant capable of improving convergence behaviour, underutilized elements including the zone changing mechanism and decreasing the prospect of premature convergence. It is very desirable to have a better balanced YSGA model that provides less computational complexity to reduce these limitations.

The proposed work aims to overcome these limitations and develop an improved YSGA framework, keeping the computational efficiencies and performance of convergence.

3. Related Terminologies

3.1 Standard Yellow Saddle Goatfish Algorithm (YSGA)

YSGA was proposed by Zaldivar et al. (2018) by imitating the hunting behaviour of Yellow Saddle Goatfish. Every fish is considered a population in this algorithm, that views the search space as the hunting region. They designated the algorithm in such a way that two searching agents or fishes i.e., chaser and blocker are used for locating the prey. For every group, only one fish becomes chaser while all other act as blocker fish automatically. Based on the category, every element is subjected to various evolutionary operations that determine the collaborative behaviour of their natural hunting. The success or efficacy of that particular element during hunting is determined by fitness function. The standard YSGA usually comprises of five phases—initial, chaser, blocker, roles exchange and finally zone changing.

***Initial stage*:** During this stage, the population $P = \{p_1, p_2, \ldots, p_m\}$, of m goatfishes is randomly generated. The population members are uniformly allocated within the allowable upper and lower limits of the search domain. Equation (1) is used for initializing the standard YSGA.

$$p_i^j = rand\left(b_j^{high} - b_j^{low}\right) + b_j^{low} \quad (1)$$

where, rand ∈ [0,1], and individual index and dimension denoted by i and j respectively.

It is important to mention here that YSGA creates groups or subsets for executing the hunting process. To achieve this, the k-means clustering approach is used in which "P" is further categorized into "k" clusters,

forming groups within spatial neighbour. It is important to note that both chaser and blocker fish are selected from the same population P, and no additional agents are introduced.

***Chaser fish*:** For each group, there is one chaser fish denoted by Φ_l which performs the hunting operation. To select the chaser fish in a particular cluster, its fitness function is evaluated. The fish with the best fitness value is chosen as the chaser because of its proximity to the solution. However, if a prey escapes the chaser fish's coral and hides in crevices, the Levy flight's random walk process is used to find the prey, as given by Equation (2).

$$\Phi_l^{(r+1)} = \Phi_l^r + \alpha \oplus Levy(\beta) \tag{2}$$

where, $0 < \beta \leq 2$.

In this equation, Φ_l^{r+1} denotes the updated position of chaser fish while Φ_l^r represents its current location with $\alpha = 1$ denoting the step size. β depicts the Levy index that controls the shape of the distribution and $\oplus$ denotes entry-wise multiplication.

***Blocker fish*:** Once the chaser fish is selected, the remaining fishes in cluster automatically become blocker fish, represented by φ_g. The main responsibility of the blocker fish is to create a protective cover around the corals during the hunting process to block escaping routes. To accomplish this, blocker fishes move in logarithmic spiral around Φ_l. The updated position of the blocker fish is presented in Equation (3).

$$\varphi_g^{(r+1)} = D_g \, e^{b\rho} \, \cos(2\pi\rho) + \Phi_l \tag{3}$$

Here, ρ is a randomly generated value within the interval [a, 1] where a is reduced gradually from −1 to −2 during the iterations to encourage stronger exploitation. The spiral's shape and direction are determined by the constant parameter b, which is defined as $b = 1$. D_g is the distance between the chaser and blocker fish's current positions within the cluster.

***Role exchange*:** This phase occurs once both the chaser and blocker fishes are selected. Since, the chaser fish is closest to prey during the hunting process and the blocker fish's objective is to avoid the prey from escaping. However, it is possible for the prey to come into the hunting area closest to blocker fish. In this case, the blocker fish leads the hunting process by becoming chaser fish and vice versa. This process is referred to as Role Exchanging.

***Zone changing*:** During the optimization process, a cluster may reach a stage where no further improvement is observed, indicating that the local search region has been sufficiently exploited. To prevent stagnation and enhance exploration, a zone changing strategy is employed. Let λ denote the maximum number of consecutive iterations allowed without improvement in the fitness value of the chaser fish within a cluster. If this limit is exceeded, the entire cluster is relocated towards a new region in the search space guided by the global best solution.

The position of each goatfish is updated as follows:

$$p_g^{(r+1)} = \frac{\Phi_{best} + p_g^r}{2} \tag{4}$$

where, p_g^{r+1}represents the updated location of the fish, Φ_{best} denotes the best solution among all clusters and p_g^r represents the current location of the agents. By balancing exploration and exploitation, this

mechanism shifts the cluster toward a more promising area. Additionally, when a better fitness value is obtained during the search process, the global best solution Φ_{best} is updated.

4. Proposed Methodology

4.1 Proposed Zone Changing Modified YSGA

In the traditional YSGA, the rearrangement of the search agents after over-exploitation is accomplished by the simple midpoint-based update of the current group position to the globally optimal solution. Despite its simplicity and efficiency, this mechanism lacks adaptability since it does not consider the past patterns of movement and fails to offer enough directional flexibility during the transition into a new search area. Consequently, the zone-changing process can perish due to less population diversity and slower convergence in case of complex or multimodal optimization landscapes. Compared with this zone changing step midpoint-based strategy, the proposed work adds the effect of the preceding movement to the zone changing step along with the controlled stochastic elements. The use of memory of movements allows creating smoother and more stable transitions, and the introduction of random terms increases the possibility of exploring new areas. This combination assists in maintaining a more efficient balance when compared with traditional YSGA.

The importance of improving YSGA's zone-changing phase is evident from the fact that the algorithm gets dual opportunities to check and locate the highly effective solutions within the given search space. The proposed algorithm follows the same steps until the role-exchanging phase; however, the zone-changing phase is now performed using a modified formulation. Before that, the conditions under which zone changes occur are discussed in following subsection.

4.1.1 System Model

The YSGA performs the zone-changing operation after it has finished exploiting a particular search area. This implies that no prey (or solution) has been located in this area, prompting the algorithm to move to other sectors of the search space to find new prey or potential solutions. During this phase, the over-exploitation factor "λ" is taken into consideration. Therefore, in each cluster, when the maximum iteration condition is exceeded and no better solution is found, a successful hunt is deemed completed, and the zone-changing operation is executed for that cluster using Equation (5).

$$p_g^{(r+1)} = \Phi_{best} + w\left(p_g^r - p_g^{r-1}\right) + r_1\left(\Phi_{best} - p_g^r\right) + r_2\gamma\left(\Phi_{best} - p_g^r\right) \quad (5)$$

Equation (5) represents a modified version of the standard YSGA, incorporating additional parameters to introduce randomness and control the influence of previous positions on the current movement of particles. The above position update equation is applied element-wise to each dimension of the solution vector. In this formulation, r_1 and r_2 are uniformly distributed random values in the range [0,1]. Although both terms are directed towards Φ_{best}, they differ in their scaling, whereas r_1 gives the main stochastic direction, while r_2 gives the controlled stochastic perturbation for diversity. In addition, p_g^{r+1} refers to the updated position of the fish, p_g^r to the current position, and p_g^{r-1} to the previous position including a memory effect that affects the current position. The best solution from all clusters is the term Φ_{best}. p_g^r and Φ_{best} are solution vectors. Furthermore, the parameters w and γ control the impact of the past movement and the stochastic perturbations to the position update. The modified formulation provides more flexibility in the zone change because it adds random variation in the zone transition with directional motion as compared to the midpoint-based update in Equation (4). This allows the agents to adjust more effectively as they navigate through a new part of the search space, while keeping a balance between global exploration and local exploitation.

The importance of using Φ_{best} is that every particle is directed to the optimal solution known by the entire population. Moreover, the zone-changing equation includes the term $w\left(p_g^r - p_g^{r-1}\right)$ to include the effect of the previous position on the current transition. In this case, "w" is a weighting factor that determines how much contribution of the last movement is being included. The larger "w" the more effect does past trajectory have on the current value, leading to smoother transitions, the smaller value the faster adaptation with less stability. This term adds a memory effect which allows for more stable and consistent movement of particles.

In addition, $r_1\left(\Phi_{best} - p_g^r\right)$ is a stochastic term that pushes the particle toward the global best position and adds randomness for exploring the search space. Similarly, the term $r_2\gamma\left(\Phi_{best} - p_g^r\right)$ introduces another controlled perturbation, with the parameter "γ" controlling the strength of this effect. This term further increases population diversity and helps prevent premature convergence to sub-optimal solutions.

The parameters w and γ determine the overall behavior of the proposed zone-changing mechanism. The parameter w is used to balance stability and adaptability, as it determines how much the stability from the previous movement should influence the current step while γ is used to modulate the stochastic variation in the search procedure. A higher value of w increases the contribution of the previous movement to the current position update, which can promote smoother transitions during the search process. In contrast, a lower value of w reduces the influence of past movements, allowing the search agents to respond more quickly to changes in the search space, although this may result in less consistent search behaviour. Likewise, larger γ values also improve the exploration ability, but may lead to slower convergence rate, and on the other hand, smaller γ values may cause early convergence. The values of w and γ are determined experimentally and set to 0.5 and 0.2, respectively, in order to get a reasonable balance between speed of convergence and quality of the solution.

To justify the selection of these parameters, a preliminary sensitivity study was carried out on representative benchmark functions using different combinations of w and γ. The examined parameter settings were chosen to reflect different levels of the memory effect and stochastic perturbation introduced in the proposed zone-changing mechanism. The results showed that, while similar best fitness values were achieved in most cases, the combination w = 0.5 and γ = 0.2 generally resulted in lower standard deviation and lower worst fitness values than the other tested settings. This behavior indicates a more stable and reliable search performance. Therefore, w = 0.5 and γ = 0.2 were adopted for all experiments presented in this study.

4.1.2 Architecture and Working

After modifying the zone-changing phase using Equation (5), the proposed work measures the fitness of each solution defined for the optimization problem. If the fitness value is less than the global best, the current fitness value replaces the global best and the global best position is updated with p_g^{r+1} as defined in Equation (5). However, if the fitness value is higher than the global best, the value of p_g^{r+1} is updated by moving it around the best position using Equation (6).

$$p_g^{(r+1)} = r_3\, \Phi_{best} \tag{6}$$

where, r_3 be any random number between [0,1] and Φ_{best} is the best position of the particle. This reinitialization strategy allows the search agents to be redistributed in the neighborhood of promising regions, which prevents premature convergence and improves the exploration capability. Thus, the

proposed approach can maintain a diversity preservation and balance between solution refinement. The detailed proposed algorithm is shown in Algorithm 1 and flow chart is depicted in **Figure 1**.

Algorithm 1: Proposed Zone Changing Modified YSGA

Input factors: N, d, IterMax, b_j^{low}, b_j^{high}, k, w, γ, Chance=0, λ=10

Final Output: Φ_{best}

Start:

Generate initial Population set P_N^d within a range of b_j^{low}, b_j^{high}
Evaluation each set to get Fitness (for all fitness functions considered individually)
Extract the global best as Φ_{best}
Split the Population P_N^d in k number of clusters
Recognize the chaser Φ_l and blocker φ_g fishes for individual cluster
Execute iterations
while (t<IterMax)
For individual cluster k
Perform hunting task for chaser fish **using Equation** (2),
Perform blocking task of blocker fish **using Equation** (3),
Calculate the fitness of individual fish & check for exchanging roles
If φ_g fitness <Φ_l fitness
Exchange roles and update Φ_l
End if
If Φ_l fitness <Φ_{best} fitness
update Φ_{best} with Φ_l fitness
End if
If Φ_l not improved
Update Chance ← Chance + 1
Else
Chance ← 0
End if
 # Main Contribution
If Chance $> \lambda$
Execute Modified change of zone process given in **Equation (5),**
Set Chance = 0
End
Calculate the fitness F_t of position p_g^{r+1} for individual fish
If F_t<Φ_{best}
Replace Φ_{best} fitness with F_t and Φ_{best} position with p_g^{r+1}
Else
Update p_g^{r+1} again with revolving it around best position using **Equation (6),**
End if
End for
$t = t$+1
End while

End

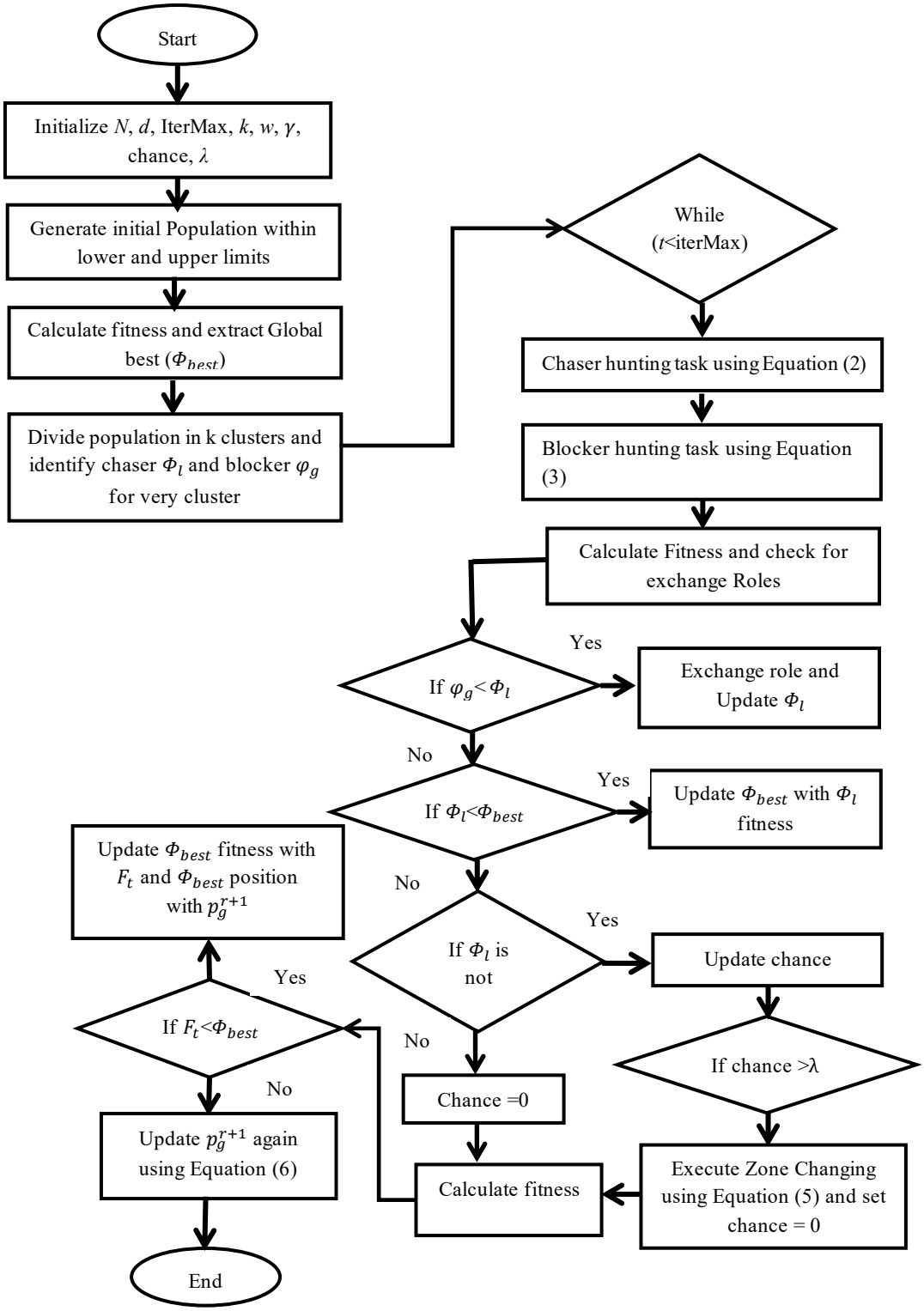


Figure 1. Highlights the modified zone change strategy introduced in the proposed YSGA.

In the proposed study, the performance of modified YSGA is evaluated using 23 standard benchmark functions widely adopted in optimization research. These benchmark functions were selected to provide a comprehensive assessment of the algorithm under diverse optimization landscapes. The details of these functions, along with their respective search ranges, are presented in **Table 2**. The selected benchmark set is designed to represent different types of optimization landscapes. Functions F1–F7 are uni-modal in nature and are used to examine the convergence behavior and exploitation capability of the algorithm. Functions F8-F13 are multimodal and have more than one local optima, therefore are appropriate to testing exploration abilities. Furthermore, F14-F23 are multimodal problems of fixed dimension, which add more complex search properties and provide an effective assessment of the robustness of the proposed algorithm.

To further examine the scalability of the modified YSGA, the variable-dimension functions are tested under varying dimensional conditions, including 50 and 100 dimensions. This enables us to validates the performance of algorithm in successively more complicated search spaces. We compared the proposed algorithm with the common optimization algorithms, such as YSGA, GWO, CS, and MFO to test its efficiency. All simulations are run in MATLAB and the respective results are discussed in the next section.

Table 2. Test functions fitness formulae and their ranges.

S. No.	Test function	Fitness equation	Range
1.	F1	$\sum_{i=1}^{n} x_i^2$	[-100. 100]
2.	F2	$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10,10]
3.	F3	$\sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	[-100. 100]
4.	F4	$\max \{ \lvert x_i \rvert, 1 \le i \le n \}$	[-100. 100]
5.	F5	$\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-30,30]
6.	F6	$\sum_{i=1}^{n} ([x_i + 0.5])^2$	[-100,100]
7.	F7	$\sum_{i=1}^{n} i.x_i^4 + random\ [0.1]$	[-1.28, 1.28]
8.	F8	$\sum_{i=1}^{n} - x_i \sin(\sqrt{x_i})$	[-500,500]
9.	F9	$\sum_{i=1}^{n} x_i^2 - 10 \cos(2\pi x_i) + 10$	[-5.12,5.12]
10.	F10	$-20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} cos(2\pi x_i) \right) + 20 + e$	[-32,32]
11.	F11	$\frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i-1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	[-600,600]
12.	F12	$\frac{\pi}{n} \{10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_1 - 1)^2 [1 + sin^2(\pi y_{i+1})] + (y_n - 1)^2\}$ $y_i = 1 + \frac{x_i + 1}{4}, u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, \quad -a < x_i < a \\ k(-x_i - a), x_i < -a \end{cases}$	[-50,50]

Table 2 continued…

13.	F13	$\frac{1}{10}\left\{sin^2(3\pi x_i) + \sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1) + (x_n-1)^2[1+sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i,5,100,4)$	[-50,50]
14.	F14	$\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	[-65,65]
15.	F15	$\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	[-5,5]
16.	F16	$4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1.x_2-4x_2^2+4x_2^4$	[-5,5]
17.	F17	$\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos(x_1)+10$	[-5,5]
18.	F18	$[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1.x_2+3x_2^2)] \times[30+(2x_1-3x_2)]^2\times(18-32x_1+12x_1^2+48x_2 - 36x_1.x_2+27x_2^2)$	[-2,2]
19.	F19	$-\sum_{i=1}^{4}c_i.\exp\left(\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right)$	[0,1]
20.	F20	$-\sum_{i=1}^{4}c_i.\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right)$	[0,1]
21.	F21	$-\sum_{i=1}^{5}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	[0,10]
22.	F22	$-\sum_{i=1}^{7}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	[0,10]
23.	F23	$-\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	[0,10]

5. Experimentation, Results and Analysis

This study, improved the standard YSGA algorithm by modifying its zone changing equation to enhance convergence rate and avoid tendency to get trapped in local minima. The hunting behaviour of YSGA algorithm makes it a better choice for real world optimization problems.

5.1 Comparative Analysis of Modified YSGA on Different Dimensions

In the first scenario, the proposed algorithm was evaluated against four traditional optimization algorithms on standard dimensions over 1000 iterations. This comparison assesses the best, worst, and standard deviation values across 23 benchmark test functions. In the following two cases, the dimensions were scaled to 50 and 100. Based on these measurements, the efficiency and dispersion of proposed algorithm can be measured using higher-dimensional problems. The 13 test functions were selected to evaluate the modified approach to offer a specific and detailed analysis of the model behaviour on a wide sample of optimization problems. The variable-dimension functions are tested under varying dimensional conditions, including 50 and 100 dimensions to prove its scalability. This will give a very strict test of the possibilities of the proposed model without going very far in carrying out the analysis than what is necessary. Such a balance between a broad and practical approach offers substantial insights into the effectiveness of the model. **Table 3** explains the simulation parameter settings of MATLAB that was used to analyze the test functions.

Table 3. Parameter settings for all compared algorithms.

Algorithms	Parameter settings
Common Settings	Population size = 30 Number of iterations = 1000 Independent runs = 30
CSA	P_a = 0.25, β = 1.5, α =1
GWO	Control parameter a = [2,0]
MFO	b = 1, t = [-1,1], spiral convergence parameter $a \in$ [-1,2]
YSGA	k = 4, Max Limit to zone change (λ) = 10
Modified YSGA	k = 4, Max Limit to zone change (λ) = 10, w = 0.5, γ = 0.2

To ensure a fair comparison, each algorithm was tested under identical experimental settings. The population size, number of trials and termination criteria were maintained same across all the methods. Each algorithm was performed 30 times independently to reduce the impact of randomness, and the average results were obtained. The parameters were determined by the values commonly used in earlier studies and were further refined by initial trial runs. The population size was determined to be N = 30, which provides a reasonable trade-off between the search performance and computational effort. Furthermore, the algorithms were also tested on higher dimensional setups (d = 50 and d = 100) to analyze their performance in more complex search spaces. The limits of the search were set based on each fitness function to make sure that the solution space was valid in the given problem.

In the modified YSGA, the grouping parameter k is used to determine how the search space is partitioned into groups and was set to k = 4, which allows enough exploration without making the computation process more complex.

To maintain search process stability, the w = 0.5 parameter estimates the effect of the previous position on the current movement. The random perturbation parameter γ = 0.2 controls the degree of random perturbation, which encourages diversity and avoids early convergence. The zone-changing limit λ = 10 was chosen to enable the algorithm to explore new areas effectively without too much or unnecessary transition.

5.1.1 Comparative Analysis of Modified YSGA on Unconstrained Test Functions with Standard Dimensions

We have examined and compared the performance of proposed algorithm with other four selected traditional optimization algorithms on 23 test benchmark functions. By using these benchmarks, we obtained the significant result that shows outperformance of the modified YSGA.

As we have mentioned previously that test functions (F1 to F7) are used for checking the exploitation abilities of the optimization algorithms while as, multimodal functions (F8 to F13) and fixed dimensions functions (F14 to F23) are used for analyzing the exploration abilities of an optimization algorithm. The statistical results obtained for proposed algorithm and other optimization algorithms are given in **Table 4** in terms of fitness, worst and standard deviation (STD) values.

Although the proposed Modified YSGA shows strong performance across many benchmark functions, it does not outperform all comparative algorithms in every case. Therefore, both the strengths and limitations of the proposed approach are discussed to ensure a balanced and transparent evaluation. As presented in **Table 4**, the proposed model achieves the best results for F1, F2, F3, F4, where the optimal value of **0** is obtained, clearly outperforming the other algorithms. For F7, the proposed approach

produces a best value of 1.67311E-05, which is considerably smaller than the values achieved by the competing methods. It is worth noting that the CS and MFO algorithms produce a worst fitness value of 0 for functions F8, F19, F20, F21, F22, and F23. Since the best values obtained for these functions are negative, a worst value of 0 indicates a noticeable difference in performance among different runs. This suggests that although these algorithms are capable of obtaining good solutions in certain executions, they do not perform with the same level of consistency throughout all runs. The observed variation suggests that the search process was not equally effective across all runs, possibly due to premature convergence or limited exploration of the search space. On the other hand, the proposed Modified YSGA shows smaller differences between its best and worst fitness values, indicating a more stable search behaviour and over repeated executions.

Table 4. Experimental results of modified YSGA with standard dimension.

Function	Parameters	YSGA	GWO	CS	MFO	Modified YSGA
F1	Best	3.9876E-126	2.48818E-68	2.76044E-13	5.41314E-21	0
	Worst	10893.63598	21470.21161	2.76044E-13	5.41314E-21	9154.793502
	STD	506.3324658	999.544297	8.72927E-14	1.71179E-21	726.0223285
F2	Best	2.05111E-69	6.98035E-38	1.56653E-08	1.29278E-14	0
	Worst	15.66536874	29.42894581	1.56653E-08	1.29278E-14	16.46648533
	STD	1.179587945	2.119890296	4.95382E-09	4.08813E-15	0.923867059
F3	Best	0.005376681	3.62018E-29	1.88025E-05	3.798202229	0
	Worst	5864.636047	38088.92903	1.88025E-05	3.798202229	21837.1113
	STD	643.2827471	1805.781467	5.94588E-06	1.201097006	1576.354693
F4	Best	9.02327E-05	4.12E-20	0.231143171	14.55227164	0
	Worst	57.89142686	84.82262896	0.231143171	14.55227164	60.7227253
	STD	6.356364968	5.952987753	0.073093888	4.601832352	3.331876469
F5	Best	2.742722498	8.05091291	4.468755285	6.859516172	6.497042313
	Worst	226.9446672	80935063.18	4.468755285	6.859516172	15528706.23
	STD	18.04918744	2653562.607	1.413144501	2.169169475	491060.536
F6	Best	2.89491E-08	2.54301E-06	2.51326E-16	1.7429E-16	0.001211569
	Worst	17223.03207	19582.22266	2.51326E-16	1.7429E-16	13025.00058
	STD	707.3528643	941.2782326	7.94763E-17	5.51154E-17	411.9201118
F7	Best	0.000225448	0.000507431	0.012596978	0.210485576	1.67311E-05
	Worst	6.272431794	10.1185332	0.012596978	0.210485576	5.037513681
	STD	0.375924648	0.413893649	0.003983514	0.066561384	0.159426947
F8	Best	-2685.929751	-2562.94518	-7150.742534	-3063.139467	-1776.000684
	Worst	-1090.552963	-1385.105164	0	0	-921.421903
	STD	237.9572942	364.9834637	2261.263337	968.6497505	98.04366097
F9	Best	1.42109E-14	0	3.31537513	48.82385452	0
	Worst	128.7246527	150.4651821	3.31537513	48.82385452	114.6644446
	STD	9.062703188	11.55664336	1.048413671	15.43945844	11.11829427
F10	Best	4.44089E-15	7.99361E-15	1.646223633	19.91598345	8.88178E-16
	Worst	19.40448403	20.6144166	1.646223633	19.91598345	17.16956513
	STD	1.745375158	2.005509402	0.520581622	6.297986955	0.559307498
F11	Best	0	0	0.050465042	0.233654756	0
	Worst	153.7887304	191.8136863	0.050465042	0.233654756	91.2380345
	STD	5.131940852	10.7009043	0.015958448	0.073888122	27.38874223
F12	Best	2.20474E-08	0.06006124	0.604865422	8.95715E-17	0.050283807
	Worst	36377084.86	52270162.81	0.604865422	8.95715E-17	30416729.89
	STD	1188350.33	1803402.936	0.191275241	2.8325E-17	961861.4517
F13	Best	0.021023858	0.191940053	1.597461589	0.010987366	0.214511148
	Worst	102223613.7	514502401.1	1.597461589	0.010987366	83587817.94
	STD	3232700.508	19033788.03	0.50516171	0.00347451	2643278.886
F14	Best	2.982105157	10.76318067	0.998003838	8.840835963	12.67050581
	Worst	12.67051118	497.2835429	0.998003838	8.840835963	18.60215192
	STD	1.626542068	15.41067264	0.315596524	2.795717806	0.187575105
F15	Best	0.00030751	0.000310686	0.000307486	0.022553327	0.000523154
	Worst	0.047998486	0.689654753	0.000307486	0.022553327	0.155730489
	STD	0.003083422	0.024143447	9.72356E-05	0.007131988	0.006672567

Table 4 continued…

F16	Best	-1.031628453	-1.031628452	-1.031628453	-1.031628453	-0.999904736
	Worst	-0.195806398	0.233214135	0	0	-0.070793797
	STD	0.034961256	0.04798678	0.326229561	0.326229561	0.04422814
F17	Best	2.48e+01	5.10e-01	1.40e+02	5.55e+00	1.44e+01
	Worst	28.03576	29.9756	27.97777	30.5638	27.95646
	STD	3.68E+01	3.44E+00	2.84e+00	3.78e+01	2.63e+01
F18	Best	3.000000007	3.00000975	3	3	3.002159214
	Worst	66.79956231	191.1534832	3	3	252.5243925
	STD	2.556303018	6.78387278	0.948683298	0.948683298	10.08072545
F19	Best	-3.862782121	-3.862686592	-3.862782148	-3.862782148	-3.862052496
	Worst	-3.829372627	-3.318687215	0	0	-3.758808876
	STD	0.001111211	0.046298175	1.221518969	1.221518969	0.01142797
F20	Best	-3.321994987	-3.202708053	-3.321995172	-3.197382698	-3.298693854
	Worst	-0.964014908	-1.467106798	0	0	-1.214132985
	STD	0.112154936	0.153751558	1.050507112	1.011101188	0.127992985
F21	Best	-10.15319962	-10.15234323	-10.15319968	-10.15319968	-5.02461434
	Worst	-1.187930933	-0.747132132	0	0	-0.664227739
	STD	1.181241628	2.740554649	3.210723652	3.210723652	0.43762672
F22	Best	-10.40294026	-10.40110387	-10.40294057	-10.40294057	-5.062622826
	Worst	-0.45245645	-0.301759481	0	0	-1.583062838
	STD	1.071444196	2.250747827	3.289698655	3.289698655	0.156661289
F23	Best	-5.128480786	-10.53519997	-10.53640982	-10.53640982	-5.117794538
	Worst	-0.419113694	-0.890509709	0	0	-0.602226755
	STD	0.251010468	2.724932794	3.331905338	3.331905338	0.323295341

In the case of function F8, the CS algorithm has the best value of -7150.742534 and this is smaller than the values of other algorithms. Nevertheless, it's worst score of 0 shows that the algorithm was successful in certain runs and failed in others. Comparatively, the proposed algorithm gets a best value of -1776.000684 and a worst value of -921.421903 with a lower variability and thus more predictable performance, although not as low as the results of CS. The proposed model also works sufficiently well with F9 and F10 where the results achieved are equal or even superior to the results of the traditional algorithms. In some instances, though, the outcomes of the proposed model are comparable to those of the currently existing approaches. As an example, the optimum value of F16 lies in -0.999904736, and it is near the observations made by YSGA and GWO. The same trend is followed with regard to the fixed-dimension functions like F18 and F19, and in this case, CS and GWO are slightly better than the proposed approach in the best fitness values. It means that more strongly locally-exploiting algorithms can be efficient in solving problems that have narrow optima or a simpler landscape.

The Modified YSGA in general shows a competitive and frequently high performance in relation to YSGA, GWO, CS and MFO. Even though, it lags to obtaining the desired results with all the benchmark functions due to complexity of the functions. These findings demonstrate the efficiency of the suggested zone-changing strategy, which would ensure to maintain the balance between exploration and exploitation and make the algorithm applicable to solving real world unimodal and multimodal optimization tasks.

5.1.2 Comparative Analysis of Modified YSGA on Unconstrained Test Functions with 50 Dimensions

The reason for increasing the dimension and examining the algorithm's best, worst, and standard deviation (STD) results in higher dimensions is that many algorithms tend to show degraded performance when solving high-dimensional problems. Therefore, it is important to evaluate the performance of the models in these more challenging environments. The statistical results obtained by each algorithm on the 13 test functions (unimodal and multimodal) with 50 dimensions are shown in **Table 5**.

Table 5. Experimental results of modified YSGA with 50 dimensions.

Function	Parameters	YSGA	GWO	Cuckoo search	MFO	Modified YSGA
F1	Best	4.81945E-77	3.83977E-27	8.71168E-15	33013.90283	0
	Worst	86819.42864	125417.0213	8.71168E-15	33013.90283	71026.52922
	STD	4521.508668	7941.974128	2.75488E-15	10439.91274	5563.165291
F2	Best	8.93251E-51	1.15539E-16	2.39216E-06	42.00116586	0
	Worst	73.94340112	1.10474E+24	2.39216E-06	42.00116586	120.5752816
	STD	5.188342794	3.49351E+22	7.56468E-07	13.28193485	6.841858705
F3	Best	1937.572046	0.010774171	0.001780548	77499.53207	0
	Worst	76068.04568	554613.2261	0.001780548	77499.53207	514470.8533
	STD	821686.4208	35843.99081	0.000563059	24507.50389	37722.31548
F4	Best	14.107313	1.77532E-05	0.239190132	93.32470835	0
	Worst	87.15756881	96.24842098	0.239190132	93.32470835	93.40917674
	STD	7.969747294	17.80040662	0.075638561	29.51186404	4.969059665
F5	Best	46.18585301	48.68814984	7.972029497	2109840.436	46.93611935
	Worst	367604536.5	664827344.1	7.972029497	2109840.436	412920762.9
	STD	19299648.61	36479709.69	2.520977078	667190.1277	13057698
F6	Best	5.23984E-05	4.894524401	2.34201E-15	11746.64233	8.53395036
	Worst	74875.66953	124477.9404	2.34201E-15	11746.64233	76068.01652
	STD	3650.67942	8691.741623	7.4061E-16	3714.614463	2405.205364
F7	Best	0.00074379	0.003590949	0.001789077	63.54557322	5.6609E-05
	Worst	276.3886781	510.4505937	0.001789077	63.54557322	310.3502049
	STD	11.12121833	22.99452151	0.000565756	20.09487466	9.884025974
F8	Best	-11207.78471	-8506.196622	-7150.742458	-11619.26229	-4392.255279
	Worst	-3098.361556	-1882.692995	0	0	-2651.391529
	STD	1215.077891	1839.856982	2261.263313	3674.333356	375.4452277
F9	Best	9.962220011	3.497048291	3.943284705	396.3479735	0
	Worst	675.3643354	802.4096834	3.943284705	396.3479735	753.7272051
	STD	55.437276	110.6851232	1.246976113	125.3362342	208.2316876
F10	Best	4.44089E-15	1.32339E-13	2.03909E-08	19.47002115	8.88178E-16
	Worst	19.9014252	21.1702529	2.03909E-08	19.47002115	20.48701152
	STD	2.225535476	3.022735085	6.44818E-09	6.156961291	0.714734175
F11	Best	0	0	0.062629527	85.34139191	0
	Worst	676.5057169	1188.03613	0.062629527	26.98731771	875.2451124
	STD	29.8382778	73.83019005	0.019805195	26.98731771	263.1047272
F12	Best	6.85054E-07	0.37254162	9.55914E-08	2197217.114	0.315782185
	Worst	1111383606	1369086130	9.55914E-08	2197217.114	789969598.2
	STD	39278640.44	79841260.33	3.02287E-08	694821.0594	24981032.04
F13	Best	0.207124199	3.184298234	3.25094E-10	17286974.91	4.968797935
	Worst	1395061413	2621005787	3.25094E-10	17286974.91	813025200
	STD	60068795.9	153786931.1	1.02804E-10	5466621.458	25710114.11

The results shown in **Table 5**, proves that the Modified YSGA works extremely well for a large number of benchmark functions. The proposed model achieves the best or near-optimal values for several functions, including F1, F2, F3, F4, F7, F9, F10, and F11. In these cases, the obtained results are either zero or very close to zero, which demonstrates the effectiveness of the proposed method in guiding the search process toward the global optimum. For function F5, the proposed algorithm shows performance comparable to the standard YSGA, as both approaches achieve similar best values. This suggests that the optimization landscape of this function may be more challenging, making further improvement difficult for all algorithms.

At the same time, the results also reveal that the proposed approach does not outperform the competing algorithms for every benchmark problem. In function F6, as an example, Cuckoo Search algorithm has a much smaller best value, which means that it has a better exploitation capacity in that specific search landscape. The same can be noted in the case of function F8 where Moth-Flame Optimization and standard YSGA are able to achieve optimal values of the best fitness as compared to the proposed model.

This type of behaviour is an indication that there are algorithms which might work better when the shape of the problem is biased towards aggressive local search mechanisms.

Moreover, the proposed model works competitively to give the best results but not the lowest best value in higher-complexity functions like F12 and F13. Specifically, Cuckoo Search algorithm achieves a very small optimum of F13, which points to its robustness in solving some multimodal optimization problems. However, the Modified YSGA has a stable behaviour and yields sensible solutions even under these circumstances which means that the proposed strategy can be applied in case the search space is more complicated.

On the whole, the outcomes indicate that the Modified YSGA can provide competitive in few cases and also outperform on a wide range of benchmark functions and especially in those which require a balance between exploration and exploitation. Even though there are algorithms with a slightly higher performance on some functions, the proposed method demonstrates some stable convergence behaviour in a series of runs. These have implicated the promise of the suggested zone-changing mechanism in the solution of complicated optimization issues and have further indicated that additional refinements, including the adaptive parameter tuning of other search approaches may be effective in future endeavours.

5.1.3 Comparative Analysis of Modified YSGA on Unconstrained Test Functions with 100 Dimensions

In the final scenario, the dimensionality of the optimization problem was set to 100 to compare the performance of the proposed algorithm with traditional optimization algorithms in high-dimensional conditions. Each function was tested with 1000 iterations to achieve an overlaying evaluation of the best, worst and standard deviation (STD) results of each algorithm. The results of this analysis are presented in **Table 6**.

Table 6. Experimental results of modified YSGA with100 dimension.

Function	Parameters	YSGA	GWO	CS	MFO	Modified YSGA
F1	Best	8.58749E-75	1.48539E-19	3.53922E-14	77359.31244	0
	Worst	156256.8968	298688.3014	3.53922E-14	77359.31244	198709.5836
	STD	8982.969525	23382.65198	1.1192E-14	24463.16255	15020.7914
F2	Best	3.91466E-60	1.17396E-11	1.41654E-07	327.440609	0
	Worst	1.7024E+26	2.10264E+49	1.41654E-07	327.440609	239.933921
	STD	5.38347E+24	6.64914E+47	4.47948E-08	103.5458123	13.53707691
F3	Best	31182.5289	133.8962304	0.001916167	172452.3368	0
	Worst	2333867.038	5187735.21	0.001916167	172452.3368	1498027.263
	STD	232445.6818	228124.0524	0.000605945	54534.2172	115914.7588
F4	Best	39.43917754	0.121438485	0.10095564	95.85463874	0
	Worst	91.64447662	96.2169609	0.10095564	95.85463874	83.8328725
	STD	7.91887909	37.470599	0.031924976	30.31189827	4.361492716
F5	Best	97.98753556	98.56195483	3.412151885	273750067.8	98.2430255
	Worst	639465402.7	1232934956	3.412151885	273750067.8	660806329.1
	STD	29953346.21	105295776.3	1.079017168	86567372.39	20896678.3
F6	Best	0.000739621	13.99396555	2.46664E-13	76101.06162	10.14668564
	Worst	198562.0867	284505.834	2.46664E-13	76101.06162	219347.4445
	STD	10653.94991	23946.78424	7.80021E-14	24065.26871	6936.011276
F7	Best	0.00040113	0.003052322	0.015264875	327.9940141	1.42825E-05
	Worst	1311.586978	1870.535551	0.015264875	327.9940141	1282.123504
	STD	79.6506531	136.6267002	0.004827177	103.7208144	40.54434826
F8	Best	-20366.40517	-13986.76118	-7150.741812	-21800.53354	-8420.87328
	Worst	-3402.365974	-2195.013778	0	0	-4236.701902
	STD	2556.862822	2817.565996	2261.263109	6893.934021	792.2018473

Table 6 continued…

F9	Best	0	1.041971376	2.176581692	1135.212139	0
	Worst	1386.127764	1695.378517	2.176581692	1135.212139	1406.373608
	STD	139.7774512	172452.3368	0.688295566	358.9855988	431.8737137
F10	Best	8.88178E-16	1.59987E-10	8.43583E-08	19.63788014	8.88178E-16
	Worst	20.03234966	21.03387366	8.43583E-08	19.63788014	20.39763226
	STD	2.256913249	3.885238809	2.66764E-08	6.210042966	0.713340501
F11	Best	0	1.11022E-16	0.070877145	660.4378109	0
	Worst	1431.608859	2709.880015	0.070877145	660.4378109	1381.035924
	STD	72.88941983	220.8160285	0.022413321	208.8487735	416.3532328
F12	Best	5.909236518	0.49649618	9.9085E-12	442350533	0.195020304
	Worst	85359214.43	3511956618	9.9085E-12	44235053	1978396795
	STD	1503956155	209330797.3	3.13334E-12	139883520.9	62562397.34
F13	Best	3.868714271	8.181537619	7.14297E-13	684760916.7	6.666321556
	Worst	2293399359	5980160851	7.14297E-13	684760916.7	3281607635
	STD	84708962.04	427812278.6	2.2588E-13	216540415	103773536.8

Table 6 shows the statistical performance of the optimization algorithms when the problem dimensionality is increased to 100. The results show that the proposed Modified YSGA maintains strong performance across several benchmark functions on a high-dimensional search space. In particular, the proposed algorithm achieves the optimal value of zero for F1, F2, F3, F4, F9, and F11, indicating its strong ability to approach the global optimum and maintain stable convergence in large search spaces. The algorithm is also good on F7 which has the lowest best value compared with all the other methods. In the case of F10, the similar best value of the proposed model is the same as the standard YSGA, and this shows similar convergence behaviour.

The results also suggest that the proposed method is not well suited to all functions. Indicatively, Cuckoo Search (CS) algorithm attains lower best values on F5, F6, F12 and F13 indicating that its search mechanism can be useful in specific problem landscapes. As well as in the case of F8, Moth-Flame Optimization (MFO) and the standard YSGA find lower best values than the proposed model.

In general, the findings indicate that the Modified YSGA offers a competitive and in most cases, better performance in high-dimensional settings in most of the test functions. In spite of the fact that certain algorithms are more effective in certain functions, the modified method ensures good convergence and equal performance in the majority of cases and it outlines the effectiveness of the presented zone-changing strategy.

5.2 Statistical Analysis

To perform a rigorous statistical evaluation of the considered optimization algorithms over the 23 benchmark functions, non-parametric statistical tests were employed. Since the obtained results do not necessarily follow a normal distribution, non-parametric methods are more appropriate for reliable comparison. Initially, the Friedman test was applied to assess the overall performance differences among the algorithms, namely Modified YSGA, CS, YSGA, GWO, and MFO. The Friedman test ranks the algorithms for each benchmark function, where rank 1 corresponds to the best (lowest) fitness value. The average ranks are then computed across all functions. The null hypothesis assumes that all algorithms perform equivalently, and any observed differences are due to random variation. In this study, the significance level was set to $\alpha = 0.05$. The p-value obtained in Friedman test is 3.6865e-09, which is much less than 0.05. Thus, the null hypothesis is rejected which means that statistically significant differences exist among the compared algorithms. The average ranks obtained are as follows: CS (1.913), GWO (2.3043), MFO (2.4783), Modified YSGA (3.7826), and YSGA (4.5217). These findings indicate

that as much as the proposed Modified YSGA has shown competitive performance, it also has not shown the best overall ranking across all benchmark functions. The Friedman test thus only implies the existence of overall differences and so a pairwise post-hoc analysis was further conducted through Wilcoxon signed-rank test to compare two variables at a time.

Wilcoxon signed-rank test is a non-parametric test that is applied to the two related samples to test the hypothesis of whether the performance of the two related samples is significantly different. It was used in the present study to compare the proposed algorithm with all of the competing methods on all of the benchmark functions. The level of significance was set to $\alpha = 0.05$, and the p-value under this level will indicate a statistically significant difference. **Table 7** shows the p-values of the Wilcoxon test of the standard benchmark functions.

Table 7. p-values obtained from Wilcoxon test on 23 functions with standard dimensions.

Function	Modified YSGA vs CS	Modified YSGA vs YSGA	Modified YSGA vs GWO	Modified YSGA vs MFO
F1	6.60e-164 <	6.65e-164 <	3.33e-165 <	5.92e-164 <
F2	3.31e-165 <	3.32e-165 <	3.33e-165 <	3.32e-165 <
F3	3.30e-165 <	3.14e-165 <	3.32e-165 <	3.32e-165 <
F4	3.30e-165 <	8.28e-169 <	3.32e-165 <	2.69e-167 <
F5	2.73e-30 <	9.75e-110 <	4.81e-01 >	2.27e-39 <
F6	1.50e-02 <	5.40e-100 <	9.47e-108 <	2.16e-39 <
F7	4.15e-179 <	8.69e-168 <	1.92e-171 <	1.36e-168 <
F8	1.14e-165 <	3.36e-165 <	2.00e-165 <	1.46e-165 <
F9	2.78e-165 <	1.67e-165 <	4.11e-61 <	3.32e-165 <
F10	3.31e-165 <	1.79e-58 <	1.82e-186 <	3.32e-165 <
F11	2.37e-165 <	5.71e-164 <	2.08e-165 <	3.31e-165 <
F12	9.62e-01 >	5.73e-115 <	1.83e-09 <	3.61e-65 <
F13	1.56e-25 <	1.80e-109 <	6.09e-131 <	8.82e-21 <
F14	1.84e-142 <	3.12e-83 <	9.50e-167 <	5.59e-173 <
F15	1.20e-32 <	3.82e-135 <	4.64e-41 <	3.74e-167 <
F16	7.55e-156 <	1.67e-161 <	2.34e-164 <	2.76e-147 <
F17	3.2E-11<	3.2e-05<	2.0e-12<	2.66e-12<
F18	9.11e-83 <	2.51e-160 <	2.14e-01 >	3.82e-99 <
F19	3.19e-161 <	3.13e-165 <	3.81e-138 <	3.32e-147 <
F20	9.27e-155 <	3.29e-165 <	2.68e-26 <	5.68e-58 <
F21	4.37e-167 <	2.40e-165 <	2.19e-165 <	2.25e-176 <
F22	2.49e-164 <	7.82e-141 <	3.38e-106 <	4.16e-167 <
F23	1.53e-159 <	4.29e-149 <	5.30e-109 <	8.55e-167 <

The symbols '<' and '>' indicate whether the p-value is less than or greater than the significance level ($\alpha = 0.05$), respectively. The value of less than '<' indicates statistically significant difference between the compared algorithms, whereas the value of greater than '>', indicates non-significant difference. Particularly, a smaller fitness value implies a higher performance and the direction of the comparison is understood in this manner.

The findings indicate that the proposed Modified YSGA results in statistically significant improvements in comparison to CS and YSGA in most of the functions. Compared to GWO and MFO, the proposed methodology also shows a positive performance in most cases, although some functions that is F5 and F18 show a negative or non-significant performance whereas in some cases (e.g., F12), the differences are not statistically significant when compared to specific methods. This means that the performance of the modified algorithm might vary between the natures of the optimization problem. It is necessary to mention that Friedman test is aimed at determining the overall ranking of algorithms, whereas Wilcoxon is based on pairwise comparison. Thus, an algorithm might fail to achieve the optimal average rank and

yet, perform statistically significantly better than a number of competing algorithms. The complementary behaviour is seen in the results obtained in this study.

Overall, the statistical findings demonstrate that the proposed Modified YSGA offers a competitive and reliable optimization performance, achieving statistically significant improvements in most cases while maintaining stable behaviour across diverse benchmark functions, despite minor variations observed in a few instances. **Tables 8, 10,** and **12** present the ranks of the p-values obtained from the Wilcoxon signed-rank test. The ranking is performed within each pairwise comparison column. Lower rank values correspond to smaller p-values and indicate stronger statistical significance for the respective pairwise comparison.

Table 8. Rank of Wilcoxon p-values when comparing pairwise (standard dimensions).

Function	Modified YSGA vs CS	Modified YSGA vs YSGA	Modified YSGA vs GWO	Modified YSGA vs MFO
F1	11	11	9	14
F2	8	8	10	13
F3	7	6	7	11
F4	6	1	8	4
F5	19	18	22	21
F6	21	20	15	20
F7	1	2	2	3
F8	3	9	4	8
F9	5	3	17	10
F10	9	22	1	12
F11	4	10	5	9
F12	22	17	20	18
F13	20	19	13	22
F14	16	21	3	2
F15	18	16	18	5
F16	14	12	11	15
F17	20	16	9	11
F18	17	13	21	17
F19	12	5	12	16
F20	15	7	19	19
F21	2	4	6	1
F22	10	15	16	6
F23	13	14	14	7

The rank values presented in **Table 8** reflect the relative strength of statistical significance among the pairwise comparisons. These ranks do not indicate the superiority of any algorithm; the actual performance comparison is based on the fitness values reported in the corresponding result tables. In addition, the same analysis was extended to higher-dimensional cases (50 and 100 dimensions). The corresponding p-values are reported in **Tables 9** and **11**, while **Tables 10** and **12** present the associated rank values for different benchmark functions.

Table 9. p-values obtained from Wilcoxon test on 13 functions with 50 dimensions.

Function	Modified YSGA vs CS	Modified YSGA vs YSGA	Modified YSGA vs GWO	Modified YSGA vs MFO
F1	6.64e-164 <	3.32e-165 <	3.33e-165 <	3.32e-165 <
F2	6.60e-164 <	3.32e-165 <	3.33e-165 <	3.32e-165 <
F3	6.61e-164 <	6.58e-164 <	3.29e-165 <	3.32e-165 <
F4	5.01e-164 <	3.33e-165 <	3.31e-165 <	6.67e-175 <
F5	2.64e-109 <	4.17e-84 <	3.19e-165 <	3.32e-165 <
F6	3.84e-22 <	2.46e-86 <	3.29e-165 <	3.32e-165 <
F7	9.49e-165 <	4.16e-166 <	7.55e-167 <	2.57e-165 <
F8	2.02e-168 <	3.46e-165 <	2.85e-29 <	2.03e-164 <

Table 9 continued…

F9	5.35e-164 <	3.16e-165 <	2.75e-165 <	3.32e-165 <
F10	3.30e-165 <	2.01e-70 <	4.78e-166 <	3.32e-165 <
F11	5.65e-164 <	1.91e-37 <	7.71e-65 <	3.32e-165 <
F12	9.75e-47 <	6.48e-85 <	3.78e-72 <	3.32e-165 <
F13	2.62e-76 <	1.43e-41 <	1.67e-165 <	3.32e-165 <

Table 10. Rank of Wilcoxon p-values when comparing pairwise (50 Dimensions).

Function	Modified YSGA vs CS	Modified YSGA vs YSGA	Modified YSGA vs GWO	Modified YSGA vs MFO
F1	9	4	9	8
F2	7	3	10	4
F3	8	7	6	5
F4	4	5	8	1
F5	10	10	5	12
F6	13	8	7	11
F7	3	1	1	2
F8	1	6	13	13
F9	5	2	4	3
F10	2	11	2	7
F11	6	13	12	6
F12	12	9	11	10
F13	11	12	3	9

Table 11. p-values obtained from Wilcoxon test on 13 functions with 100 dimensions.

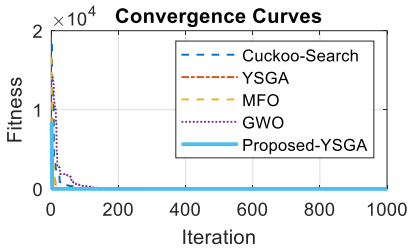
Function	Modified YSGA vs CS	Modified YSGA vs YSGA	Modified YSGA vs GWO	Modified YSGA vs MFO
F1	6.64e-164 <	3.32e-165 <	3.33e-165 <	3.32e-165 <
F2	3.29e-165 <	3.32e-165 <	3.33e-165 <	3.32e-165 <
F3	6.60e-164 <	1.95e-164 <	6.01e-164 <	3.32e-165 <
F4	4.92e-164 <	3.34e-165 <	2.50e-165 <	1.55e-168 <
F5	3.01e-40 <	4.53e-53 <	3.26e-10 <	3.32e-165 <
F6	3.32e-25 <	9.04e-105 <	3.98e-46 <	3.32e-165 <
F7	4.61e-165 <	1.36e-173 <	5.57e-167 <	3.32e-165 <
F8	4.03e-166 <	8.04e-165 <	8.49e-25 <	1.19e-163 <
F9	1.67e-161 <	1.84e-45 <	2.94e-165 <	3.32e-165 <
F10	3.30e-165 <	4.87e-166 <	3.33e-165 <	3.32e-165 <
F11	5.08e-164 <	3.85e-40 <	1.10e-105 <	3.32e-165 <
F12	3.93e-05 <	5.91e-164 <	3.30e-165 <	3.32e-165 <
F13	1.40e-51 <	6.21e-06 <	3.18e-13 <	3.32e-165 <

Table 12. Rank of Wilcoxon p-values when comparing pairwise (100 dimensions).

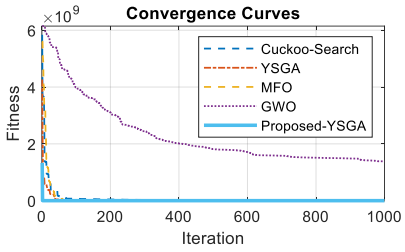
Function	Modified YSGA vs CS	Modified YSGA vs YSGA	Modified YSGA vs GWO	Modified YSGA vs MFO
F1	8	3	6	6
F2	2	4	7	3
F3	7	7	8	4
F4	5	5	2	1
F5	11	10	13	11
F6	12	9	10	12
F7	4	1	1	2
F8	1	6	11	13
F9	9	11	3	5
F10	3	2	5	7
F11	6	12	9	8
F12	13	8	4	9
F13	10	13	12	10

5.3 Convergence Analysis

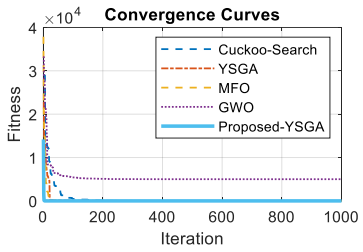
Since, one of the major challenges faced by optimization algorithms is their tendency to converge prematurely to local optima. To ensure convergence towards global optimum solution it is extremely important to maintain an effective balance between exploration and exploitation phases of algorithm. This gap is fulfilled by maintaining the equilibrium in modified YSGA by updating the zone changing phase of standard YSGA. The convergence curves obtained for different functions with standard dimensions, 50 dimensions and 100 dimensions are shown in **Figures 2, 3** and **4** respectively.



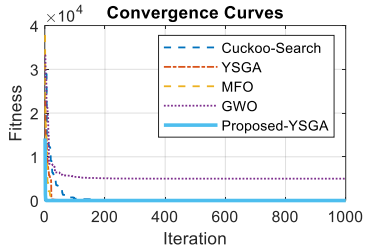
(F1)



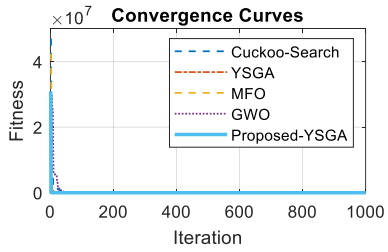
(F2)



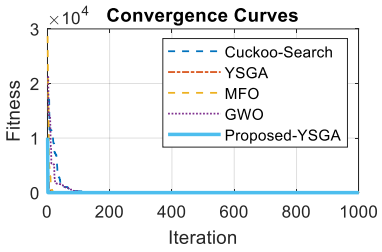
(F3)



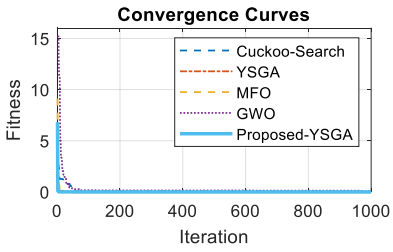
(F4)



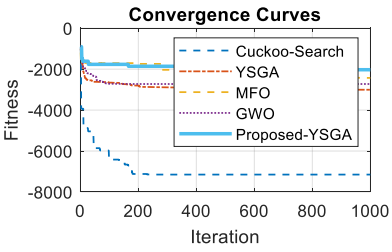
(F5)



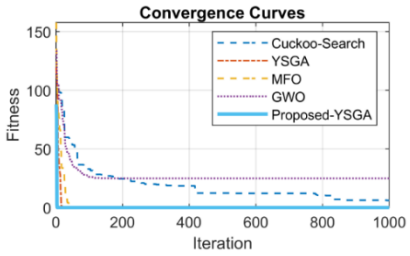
(F6)



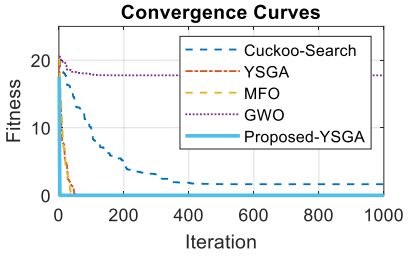
(F7)



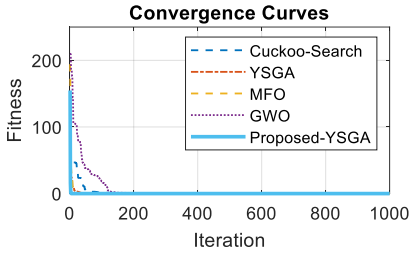
(F8)



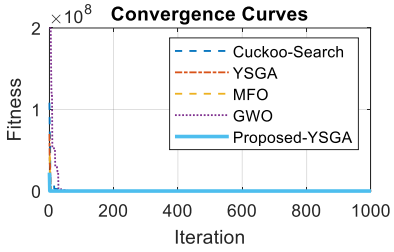
(F9)



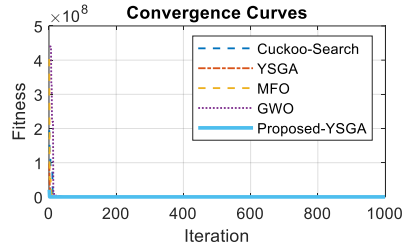
(F10)



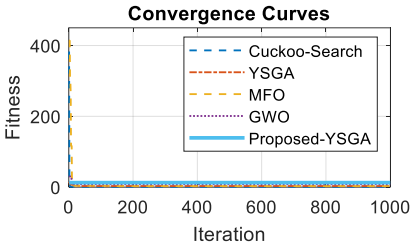
(F11)



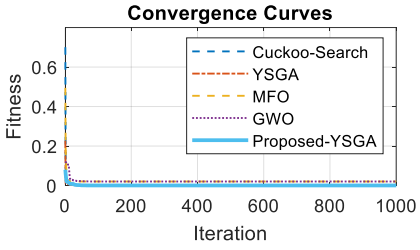
(F12)



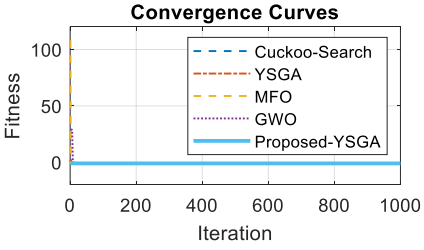
(F13)



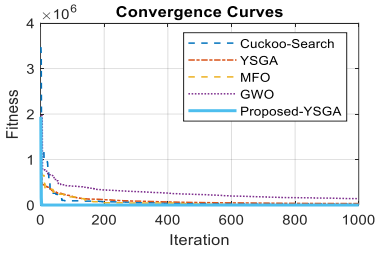
(F14)



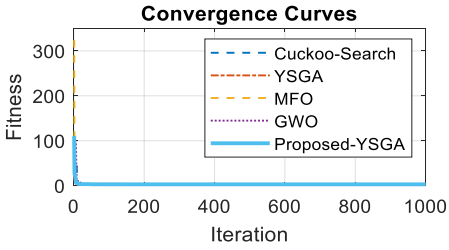
(F15)



(F16)



(F17)



(F18)

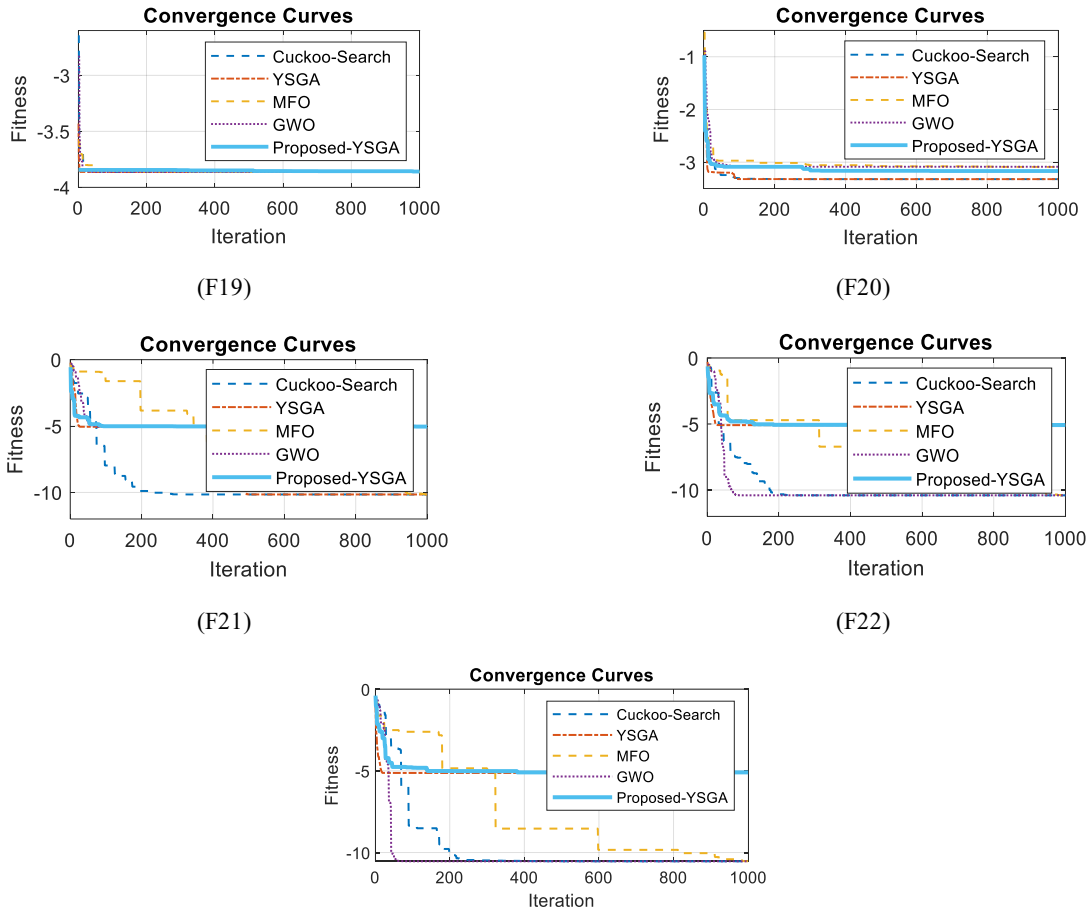
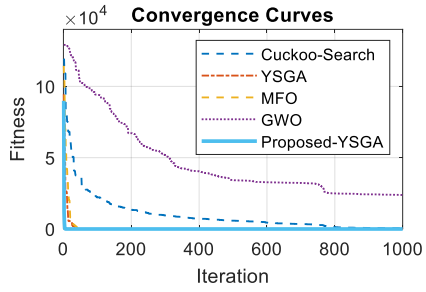
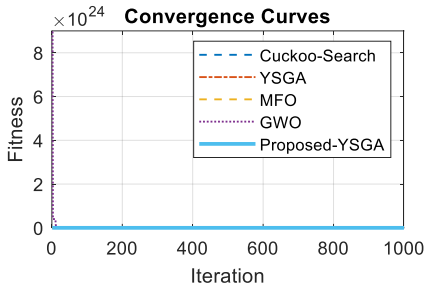


Figure 2. Convergence analysis of 23 test functions with standard dimensions.

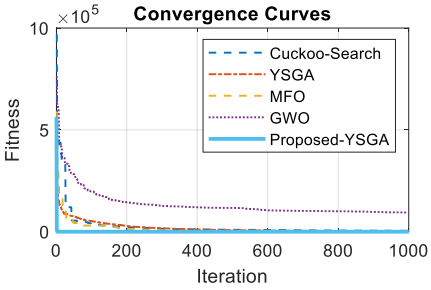
From these graphs, it is evident that the convergence curve of the proposed Modified YSGA demonstrates competitive behavior compared with the traditional algorithms. For unimodal functions (F1 to F7), the Modified YSGA achieves the best convergence rate after only a few iterations. Similar results are observed for multimodal functions (F9 to F13), except for function F8, where GWO shows the best convergence rate. For fixed-dimensional multimodal functions, the Modified YSGA performs best on functions F14 to F20. However, for functions F21, F22, and F23, the convergence curve of the Modified YSGA is slower than that of GWO but better than the other algorithms.



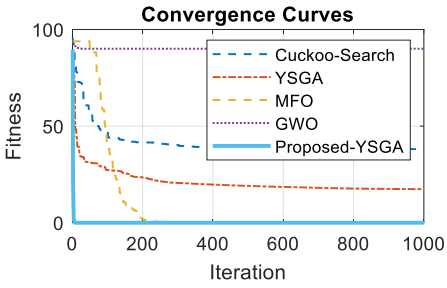
(F1)



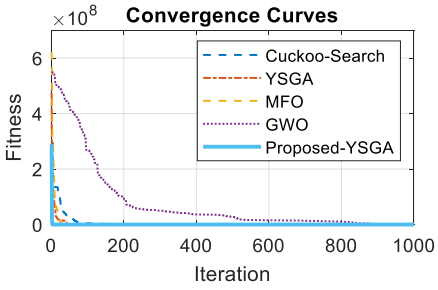
(F2)



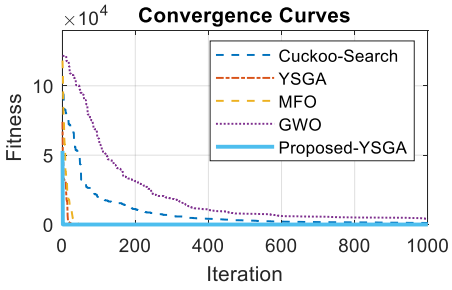
(F3)



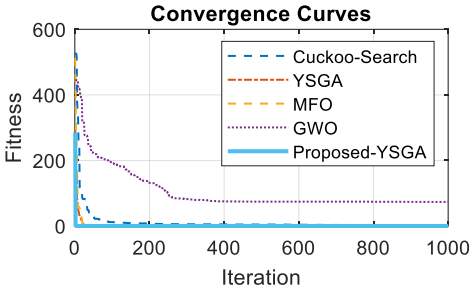
(F4)



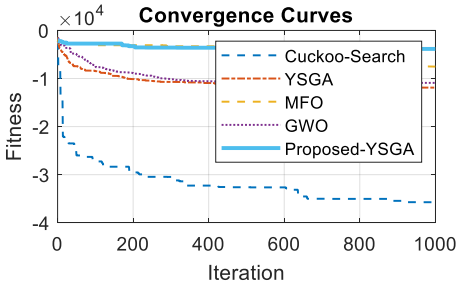
(F5)



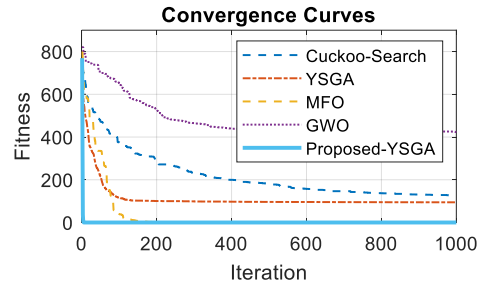
(F6)



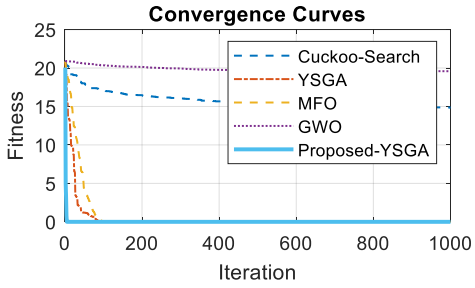
(F7)



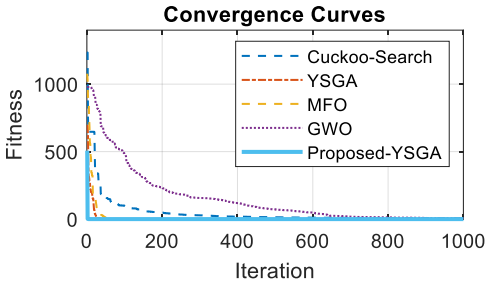
(F8)



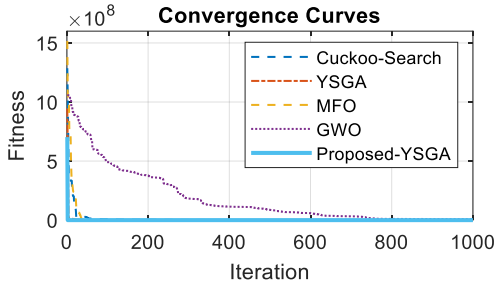
(F9)



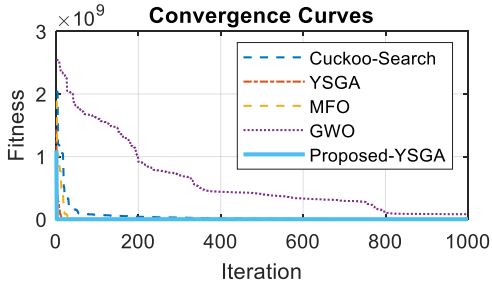
(F10)



(F11)



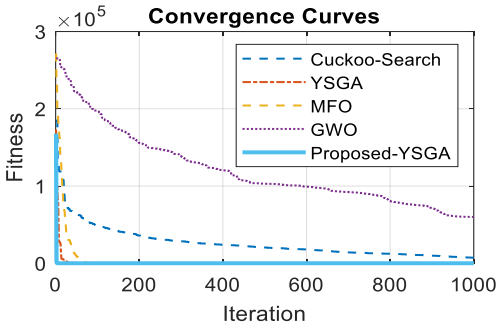
(F12)



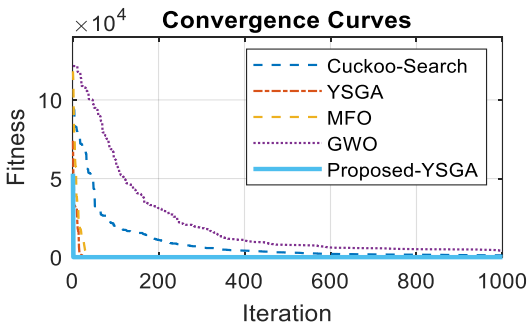
(F13)

Figure 3. Convergence analysis of 13 test functions with 50 dimensions.

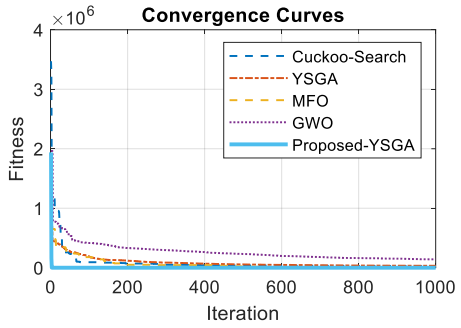
The convergence curves indicate that the proposed Modified YSGA shows competitive performance compared with conventional algorithms, particularly for the 50-dimensional unimodal (F1–F7) and multimodal (F8–F13) benchmark functions. The algorithm demonstrates fast convergence for most functions, showing its ability to reach near-optimal solutions within fewer iterations. Although slightly slower convergence is observed for function F8, the overall results confirm the effectiveness of the proposed search strategy. This behavior reflects a balanced exploration and exploitation mechanism, enabling efficient progress toward optimal solutions.



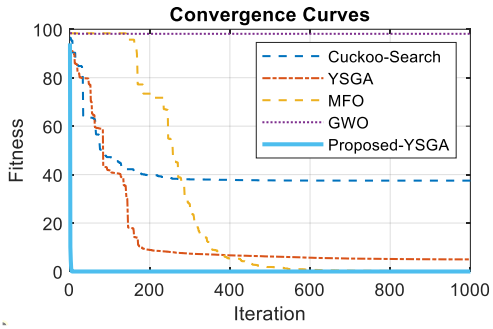
(F1)



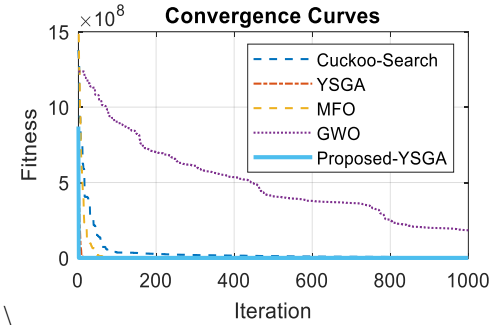
(F2)



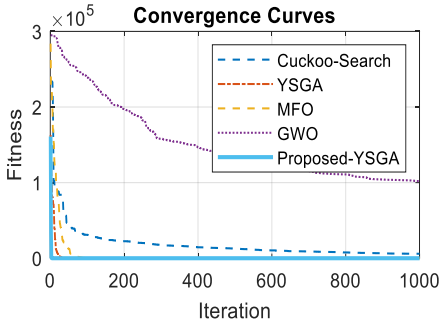
(F3)



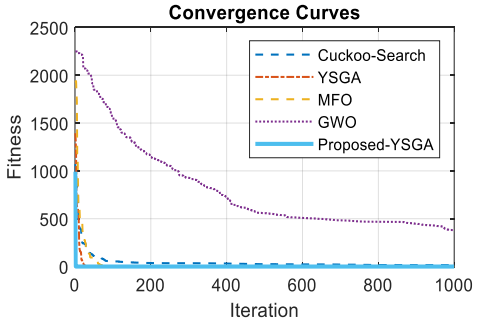
(F4)



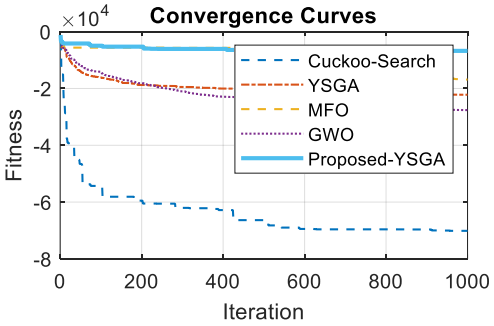
(F5)



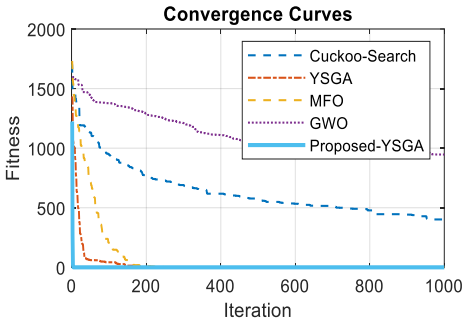
(F6)



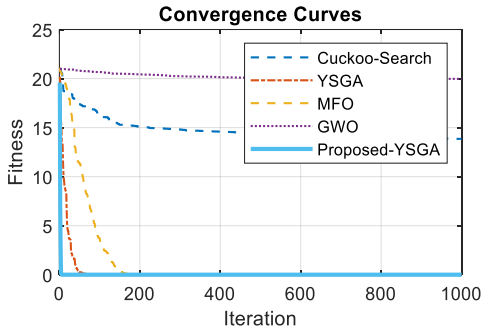
(F7)



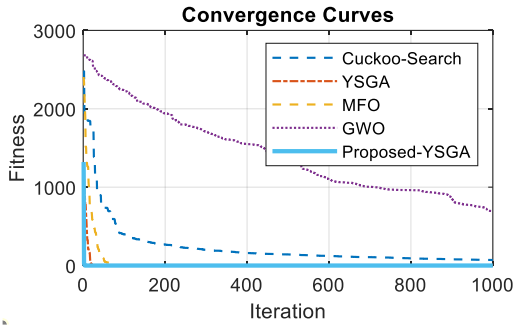
(F8)



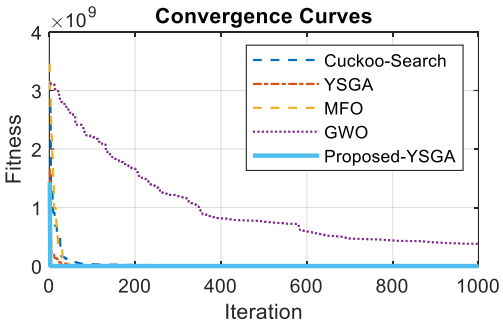
(F9)



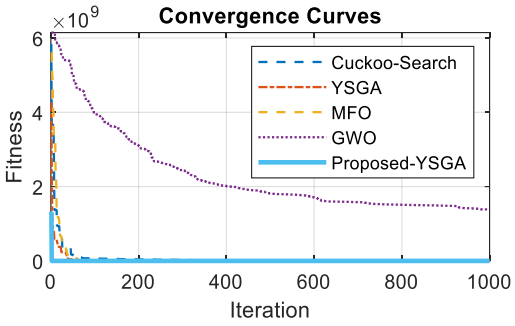
(F10)



(F11)



(F12)



(F13)

Figure 4. Convergence analysis of 13 test functions with 100 dimensions.

The convergence curves indicate that the proposed Modified YSGA attains the highest convergence rates in most test function. In the case of unimodal functions F1 to F7, with a single global optimum, Modified YSGA is unsurpassed in quickly finding the optimal solution in comparison with conventional algorithms like CS, YSGA, GWO and MFO. Likewise with the more complicated multimodal functions F8 to F13, which possess more than one local optimum, the proposed model is able to reach the highest convergence rates as well. These functions are especially difficult with the existence of many peaks and valleys of which most optimization algorithms can be misled. However, these are the strengths of Modified YSGA where it demonstrates its powerful exploration and exploitation capabilities, which allow it to perform efficiently.

6. Engineering Design Problem

This section describes proposed algorithm's performance in regard to its capability to explore and exploit real-world optimization challenges. The algorithm's practical performance can be determined by how effectively it balances exploration and exploitation and how well it searches the solution space. Various engineering applications are used to check the feasibility and strength in case of realistic conditions and limitations. To further examine the impact of the proposed approach, some classical engineering design problems are discussed, such as the design of three bars truss, pressure vessel design, and problem of spring design. To validate the efficiency and reliability in practical use of the proposed algorithm we compared the results with conventional methods.

6.1 Constraint Handling Strategy

In order to achieve safety, structural integrity, and feasibility, engineering design problems typically involve a set of inequality constraints that must be satisfied. To deal with these constraints, a fixed penalty term is applied in this study. The constrained optimization problem is transformed into an unconstrained one by adding a penalty term into the objective function where constraint violations are present (Lin, 2013).

The penalized fitness function is formulated as:

$$F(x)= f(x)+ \lambda_1 \sum_{i=1}^{m} max(0, g_i(x))^2 \quad (7)$$

where, $f(x)$ is the original objective and $g_i(x)$ is the inequality constraint function, and m is the number of constraints to be considered in the problem and λ_1 is the penalty coefficient. Only constrained violations contribute to the penalty value, but no feasible solutions contribute to the penalty value.

The penalty parameter is selected as a sufficiently large constant and is set to $\lambda_1=10^5$ based on preliminary experimentation. This option promotes sufficient balance between constraint enforcement and search capacity, which allows the algorithm to converge the possible and optimal design solutions. In boundary, constraints are managed by a simple correction process to preserve decision variables in the range in which they are allowed.

The same constraint management approach is utilized in all engineering design issues considered in this section so as to ensure fairness and uniform performance evaluation.

6.2 Three –Bar Truss

The three bar truss design problem is a classical design problem (Ray & Saini, 2001) and its structure is as shown in **Figure 5**. The aim is to reduce the truss structure's weight while maintaining stress limits, buckling, and deflections. The mathematical formulation of the problem is presented in Equation (8) and the results of the parameter optimization of each algorithm are shown in **Table 13**.

Min $f(x) = (2\sqrt{2}x_1+x_2)\times L$

Constraints:

$$g_1 = \frac{\sqrt{2}x_1+x_2}{\sqrt{2}x_1^2+2x_1x_2} P - \sigma \leq 0,$$

$$g_2 = \frac{x_2}{\sqrt{2}x_1^2+2\,x_1x_2} P - \sigma \leq 0,$$

$$g_3 = \frac{1}{x_1+\sqrt{2}x_2} P - \sigma \leq 0,$$

where, $0 \leq x_1, x_2 \leq 1$, L = 100 cm, P (Applied Load) = 2 kN /cm^2, σ = 2 kN /cm^2 (8)

To represent these inequality constraints in the optimization process, the constraint handling mechanism as presented in Section 6.1 is used. The **Table 13** summarises the results that the various algorithms yield. The calculations of the proposed algorithm give an optimal design variable of $x_1 = 0.7882$ and $x_2 = 0.4094$, with the minimum structural volume of 263.8.

When comparing the results to the other currently existing optimization methods, it is possible to note that the proposed method will give better performance in terms of quality of the solutions and the convergence behaviour. The solution acquired meets all the given constraints, which proves effectiveness and strength of the chosen optimization strategy to constrained structural design problems.

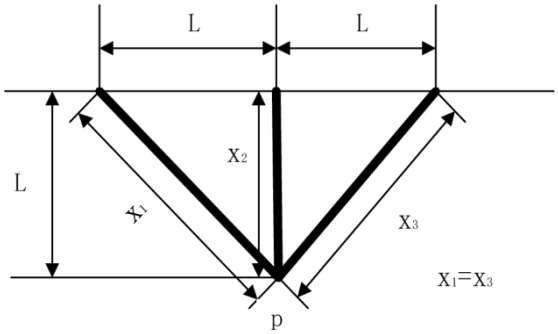


Figure 5. Three-bar truss design problem (Ray & Saini, 2001).

Table 13. Comparison of three-bar truss problem with other algorithms.

ALGORITHM	x_1	x_2	FITNESS
CSA	0.7887	0.4082	263.9
GWO	0.8010	0.3800	264.5
MFO	0.7925	0.3982	263.99
YSGA	0.8032	0.3706	264.27
Modified YSGA	0.7882	0.4094	**263.8**

6.3 Pressure Vessel Design Problem

This is another well-known engineering design problem introduced by Kannan & Kramer (1994). The thickness of the shell (Ts), the thickness of the head (Th), the inner radius of the cylinder (R), and the length of the cylinder (L) are the four variables. The aim is to reducing the manufacturing cost of the pressure vessel as depicted in **Figure 6**. Three linear and one nonlinear are the four major constraints. The mathematical statement of this problem is defined below:

Consider $x = [x_1\ x_2\ x_3\ x_4] = [T_s\ T_h\ R\ L]$

Minimize $f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$,

Constraints:

$g_1(x) = -x_1 + 0.0193x_3 \leq 0$,

$g_2(x) = -x_3 + 0.00954x_3 \leq 0$,

$g_3(x) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0$,

$g_4(x) = x_4 - 240 \leq 0,$

variable range $0 \leq x_1 \leq 99,\ 0 \leq x_2 \leq 99,\ 10 \leq x_3 \leq 200,\ 10 \leq x_4 \leq 200$ (9)

The inequality constraints are incorporated into the optimization process using the penalty-based constraint handling strategy described in Section 6.1. The four primary variables of the pressure vessel problem are optimized using the proposed approach and determine the optimal values. The algorithm achieved the lowest possible cost by fine tuning the variables while maintaining compliance with all constraints. The findings of using the proposed approach are contrasted with the outcome of five different techniques that have previously been used to address the same problem. The lowest cost values obtained by each algorithm, along with the correlating values of the optimized variables are analysed as shown in **Table 14**.

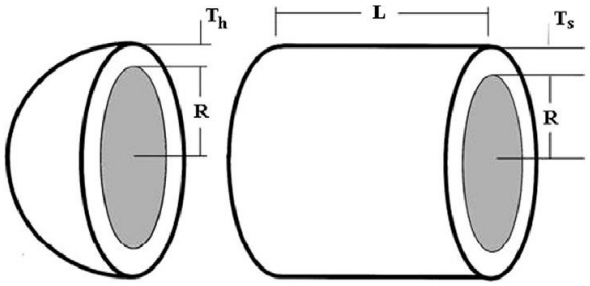


Figure 6. Pressure vessel design problem (Kannan & Kramer,1994).

Table 14. Comparison of pressure vessel design problem with other algorithms.

ALGORITHM	x_1	x_2	x_3	x_4	FITNESS
CSA	0.8125	0.4375	42.0900	176.70	6061.0
GWO	0.8125	0.4345	42.0890	176.75	6051.5
MFO	0.8125	0.4375	42.1030	176.57	6059.0
YSGA	0.7780	0.3840	40.4000	199.00	5873.0
Modified YSGA	0.7770	0.3830	40.2000	199.00	**5831.1**

The results in **Table 14** shows that the proposed algorithm is reliable and able to produce superior or comparative results to the other methods, also proves that it's robustness and effectiveness. The fact that it can find lower-cost solutions despite the constraints of the problem means that the algorithm is highly competitive. These results confirm the proposed method as emerge as a positive alternative among engineers and researchers engaged in the same issues of designing optimization.

6.4 Spring Design Problem

The spring design optimization problem is a popular benchmark problem in the optimization of engineering research because of its practical application in the designing of mechanical systems (Tzanetos & Blondin, 2023). This problem aims at reducing the weight of helical compression spring and meeting the requirements based on shear stress, deflection, surge frequency, and geometric limitations. These limitations guarantee the operation of the designed spring as a safe and reliable one at specified loading conditions. The spring design problem can be mathematically formulated in Equation (10) and the design variables are the diameter of the wire, the average diameter of the coils, and the number of coils to be used. The mathematical formulation of the issue is as follows:

Consider $x = [x_1, x_2, x_3] = [d, D, N]$

Minimize: $f(x) = (x_3 + 2)\, x_2\, x_1^2$

Constraints:

$$g_1(x) = 1 - \frac{(x_2^3\, x_3)}{(71785\, x_1^4)} \leq 0,$$

$$g_2(x) = \frac{4x_2^2 - x_1 x_2}{12566(x_1^3 x_2 - x_{1)}^4} + \frac{1}{5108 x_1^2} - 1 \leq 0,$$

$$g_3(x) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0,$$

$$g_4(x) = \frac{(x_1 + x_2)}{1.5} - 1 \leq 0,$$

where, $0.05 \leq x_1 \leq 2$, $0.25 \leq x_2 \leq 1.30$, $2 \leq x_3 \leq 15$ (10)

A spring design optimization problem is using penalty-based constraint management method as explained in Section 6.1. The effectiveness of the proposed method is compared to find out its performance through the result comparison with five established optimization algorithms. The values of the minimum objective functions attained by each method and the optimal design variables are detailed in **Table 15** and a structure depicted in **Figure 7**. This comparative analysis helps to identify the strengths and limitations of each of the optimization techniques in the effective solution of constrained spring design problem.

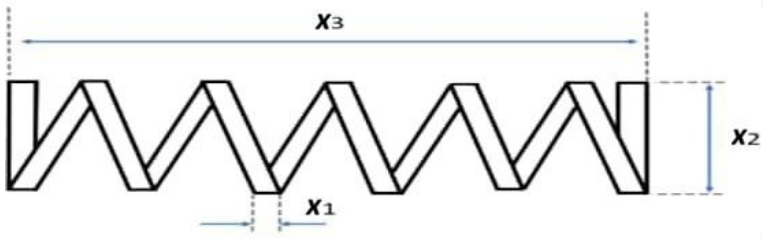


Figure 7. Spring design problem (Tzanetos & Blondin, 2023).

Table 15. Comparison of spring design problem with other algorithms.

ALGORITHM	x_1	x_2	x_3	FITNESS
CSA	0.063697	0.598020	5.756348	0.018820
GWO	0.068145	0.887019	2.273402	0.017602
MFO	0.072172	0.789945	10.499205	0.051430
YSGA	0.056077	0.471737	7.052920	0.013429
Modified YSGA	0.051422	0.350322	11.674041	**0.012667**

The performance of the proposed algorithm and other competing optimization techniques of the spring design problem are clearly compared using numerical data contained. As shown in **Table 15**, the proposed approach has the lowest objective function value meaning that it is a more efficient spring design in terms of the weight minimization and by fulfilling all the imposed constraints.

7. Conclusion and Future Scope

In this study, a modified Yellow Saddle Goatfish Algorithm was proposed to overcome the limitations of the original method, particularly low convergence rate and the possibility of getting trapped in local optima. A zone-changing mechanism was defined to improve the balance between exploration and exploitation during the search process. Unlike the standard midpoint-based strategy, the proposed modification allows more flexible transitions between search regions, which contribute to improved adaptability of the proposed algorithm. The efficiency of this approach was tested on the basis of a number of benchmark functions in various dimensional settings.

The results obtained have shown that the modified algorithm tends to converge faster and provide more accurate solutions compared to the standard YSGA and a number of widely used metaheuristic algorithms. The improvements were seen on the various categories of benchmark functions. Moreover, the Friedman rank test and Wilcoxon signed-rank test were employed to confirm the results, which prove that the improvement of the performance is statistically significant.

The findings are

- The introduced zone-changing mechanism improves convergence speed without compromising solution stability.
- Both unimodal and multimodal benchmark functions are obtained with stable and reliable results.
- The modified YSGA works well in higher-dimensional problems (50 and 100 dimensions).
- Statistical testing is used to establish the significance of the improvements.
- The method validates competitive or superior performance compared to Cuckoo Search, Moth-Flame Optimization, Grey Wolf Optimizer, and the original YSGA.

The algorithm was also used to analyse its practical applicability by applying it to constrained engineering design problems, including the three-bar truss, pressure vessel, and spring design problems. The findings indicate that the approach is feasible to manage constraints and produce viable solutions of desirable quality. This implies that the modified YSGA may also be applied to the real world optimization problems in engineering.

Despite these promising results, the present work primarily concentrates on benchmark and static optimization problems. Future studies can investigate the applicability of the modified YSGA to dynamic or large-sized real-world optimization problems, and future developments will be possible by integrating adaptive parameter adjustment or hybrid optimization methods.

Conflicts of Interest

The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgments

The authors gratefully acknowledge the support provided by the Department of Mathematics Chandigarh University, for facilitating this research work.

AI Disclosure

The author(s) declare that no assistance is taken from generative AI to write this article.

References

Brahma, D., Swayamsiddha, S., & Panda, G. (2022). Efficient spectrum allocation in cognitive radio using yellow saddle goatfish algorithm. In *2022 IEEE International Conference on Artificial Intelligence in Engineering and Technology* (pp. 1-6). IEEE. Kota Kinabalu, Malaysia. https://doi.org/10.1109/IICAIET55139.2022.9936873

Dehghani, M., Trojovská, E., & Trojovský, P. (2022). A new human-based metaheuristic algorithm for solving optimization problems on the base of simulation of driving training process. *Scientific Reports, 12*(1), 9924. https://doi.org/10.1038/s41598-022-14225-7

Ding, J., Wang, Q., Zhang, Q., Ye, Q., & Ma, Y. (2019). A hybrid particle swarm optimization–cuckoo search algorithm and its engineering applications. *Mathematical Problems in Engineering, 2019*(1), 5213759. https://doi.org/10.1155/2019/5213759

Erol, O.K., & Eksin, I. (2006). A new optimization method: big bang–big crunch. *Advances in Engineering Software, 37*(2), 106-111. https://doi.org/10.1016/j.advengsoft.2005.04.005

Gandomi, A.H., & Alavi, A.H. (2012). Krill herd: a new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation, 17*(12), 4831-4845. https://doi.org/10.1016/j.cnsns.2012.05.010

Gharebaghi, S.A., Kaveh, A., & Asl, M.A. (2017). A new meta-heuristic optimization algorithm using star graph. *Smart Structures and Systems, 20*(1), 99-114. https://doi.org/10.12989/sss.2017.20.1.099

Gharehchopogh, F.S., Nadimi-Shahraki, M.H., Barshandeh, S., Abdollahzadeh, B., & Zamani, H. (2023). CQFFA: a chaotic quasi-oppositional farmland fertility algorithm for solving engineering optimization problems. *Journal of Bionic Engineering, 20*(1), 158-183. https://doi.org/10.1007/s42235-022-00269-4

Glover, F. (1989). Tabu search—Part I. *ORSA Journal on Computing, 1*(3), 135-206. https://doi.org/10.1287/ijoc.1.3.190

Goel, A., Wasim, J., Srivastava, P.K., Malik, K., & Singh, M. (2023). Image fusion techniques based on optimization algorithms: a review. *Engineering Proceedings, 59*(1), 225. https://doi.org/10.3390/engproc2023059225

Hashim, F.A., Mostafa, R.R., Khurma, R.A., Qaddoura, R., & Castillo, P.A. (2024). A new approach for solving global optimization and engineering problems based on modified sea horse optimizer. *Journal of Computational Design and Engineering, 11*(1), 73-98. https://doi.org/10.1093/jcde/qwae001

Holland, J.H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press. London, England.

Hosseinalipour, A., Ghanbarzadeh, R., Arasteh, B., Gharehchopogh, F.S., & Mirjalili, S. (2024). A metaheuristic approach based on coronavirus herd immunity optimiser for breast cancer diagnosis. *Cluster Computing, 27*, 9451-9475. https://doi.org/10.1007/s10586-024-04360-3

Jia, H., & Lu, C. (2024). Guided learning strategy: a novel update mechanism for metaheuristic algorithms design and improvement. *Knowledge-Based Systems, 286*, 111402. https://doi.org/10.1016/j.knosys.2024.111402

Joshi, A.S., Kulkarni, O., Kakandikar, G.M., & Nandedkar, V.M. (2017). Cuckoo search optimization: a review. *Materials Today: Proceedings, 4*(8), 7262-7269. https://doi.org/10.1016/j.matpr.2017.07.055

Kannan, B.K., & Kramer, S.N. (1994). An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design, 116*(2), 405-411. https://doi.org/10.1115/1.2919393

Kashyap, D., Singh, B., & Kaur, M. (2021). Chaotic approach for improving global optimization in yellow saddle goatfish. *Engineering Reports, 3*(9), e12381. https://doi.org/10.1002/eng2.12381

Kaveh, A., & Hosseini, P. (2014). A simplified dolphin echolocation optimization method for optimum design of trusses. *International Journal of Optimization in Civil Engineering, 4*(3), 381-397.

Kaveh, A., & Khayatazad, M. (2012). A new meta-heuristic method: ray optimization. *Computers & Structures, 112-113*, 283-294. https://doi.org/10.1016/j.compstruc.2012.09.003

Kaveh, A., & Mahdavi, V.R. (2014). Colliding bodies optimization: a novel meta-heuristic method. *Computers & Structures, 139*, 18-27. https://doi.org/10.1016/j.compstruc.2014.04.005

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 – International Conference on Neural Networks* (pp. 1942-1948). IEEE, Perth, Western Australia, Australia. https://doi.org/10.1109/ICNN.1995.488968

Kusuma, P.D., & Purboyo, T.W. (2025). Adaptive iteration algorithm: a metaheuristic and its implementation on economic emission dispatch problem. *Mathematical Modelling of Engineering Problems, 12*(7), 2362-2372. https://doi.org/10.18280/mmep.120716

Leifu, G., & Xuejiao, R. (2021). Improved YSGA algorithm combining declining strategy and fuch chaotic mechanism. *Journal of Frontiers of Computer Science & Technology, 15*(3), 564-576. https://doi.org/10.3778/j.issn.1673-9418.2004036

Leung, K.H.B., Yousefi, N., Chan, T.C.Y., & Bayoumi, A.M. (2023). Constrained optimization for decision making in health care using Python: a tutorial. *Medical Decision Making, 43*(7-8), 760-773. https://doi.org/10.1177/0272989X231188027

Lin, C.-H. (2013). A rough penalty genetic algorithm for constrained optimization. *Information Sciences, 241*, 119-137. https://doi.org/10.1016/j.ins.2013.04.001

Lu, C., Gao, L., Li, X., Hu, C., Yan, X., & Gong, W. (2020). Chaotic-based grey wolf optimizer for numerical and engineering optimization problems. *Memetic Computing, 12*, 371-398. https://doi.org/10.1007/s12293-020-00313-6

Mirjalili, S. (2015). Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228-249. https://doi.org/10.1016/j.knosys.2015.07.006

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51-67. https://doi.org/10.1016/j.advengsoft.2016.01.008

Mirjalili, S., Mirjalili, S.M., & Lewis, A. (2014a). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Mirjalili, S., Wang, G.-G., & Coelho, L.d.S. (2014b). Binary optimization using hybrid particle swarm optimization and gravitational search algorithm. *Neural Computing and Applications, 25*(6), 1423-1435. https://doi.org/10.1007/s00521-014-1629-6

Mohammadi, A., & Sheikholeslam, F. (2023). Intelligent optimization: Literature review and state-of-the-art algorithms (1965–2022). *Engineering Applications of Artificial Intelligence, 126*(Part D), 106959. https://doi.org/10.1016/j.engappai.2023.106959

Mousavirad, S.J., & Ebrahimpour-Komleh, H. (2017). Human mental search: a new population-based metaheuristic optimization algorithm. *Applied Intelligence, 47*(3), 850-887. https://doi.org/10.1007/s10489-017-0903-6

Naik, M.K., Samantaray, L., & Panda, R. (2016). A hybrid CS–GSA algorithm for optimization. In: Bhattacharyya, S., Dutta, P., Chakraborty, S. (eds) *Hybrid Soft Computing Approaches*. Springer India, New Delhi, pp. 3-35. https://doi.org/10.1007/978-81-322-2544-7_1

Nayak, S.K., Senapati, B.R., & Mishra, D. (2024). Balancing exploration and exploitation: unleashing the adaptive power of automatic cuckoo search for meta-heuristic optimization. *Intelligent Decision Technologies, 18*(1), 485-508. https://journals.sagepub.com/doi/full/10.3233/IDT-idt230275

Rao, R.V., Savsani, V.J., & Vakharia, D.P. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design, 43*(3), 303-315. https://doi.org/10.1016/j.cad.2010.12.015

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information Sciences, 179*(13), 2232-2248. https://doi.org/10.1016/j.ins.2009.03.004

Ray, T., & Saini, P. (2001). Engineering design optimization using a swarm with intelligent information sharing among individuals. *Engineering Optimization, 33*(6), 735-748. https://doi.org/10.1080/03052150108940941

Rodríguez, A., Alejo-Reyes, A., Cuevas, E., Beltran-Carbajal, F., & Rosas-Caro, J.C. (2020). An evolutionary algorithm-based PWM strategy for a hybrid power converter. *Mathematics, 8*(8), 1247. https://doi.org/10.3390/math8081247

Salgotra, R., Sharma, P., Raju, S., & Gandomi, A.H. (2024). A contemporary systematic review on meta-heuristic optimization algorithms with their MATLAB and Python code reference. *Archives of Computational Methods in Engineering, 31*(3), 1749-1822. https://doi.org/10.1007/s11831-023-10030-1

Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: theory and application. *Advances in Engineering Software, 105*, 30-47. https://doi.org/10.1016/j.advengsoft.2017.01.004

Sergeyev, Y.D., Kvasov, D.E., & Mukhametzhanov, M.S. (2018). On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. *Scientific Reports, 8*(1), 453. https://doi.org/10.1038/s41598-017-18940-4

Sharma, S., Khodadadi, N., Saha, A.K., Gharehchopogh, F.S., & Mirjalili, S. (2023). Non-dominated sorting advanced butterfly optimization algorithm for multi-objective problems. *Journal of Bionic Engineering, 20*(2), 819-843. https://doi.org/10.1007/s42235-022-00288-9

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*(6), 702-713. https://doi.org/10.1109/TEVC.2008.919004

Storn, R., & Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341-359. https://doi.org/10.1023/A:1008202821328

Tawhid, M.A., & Ali, A.F. (2018). An effective hybrid firefly algorithm with the cuckoo search for engineering optimization problems. *Mathematical Foundations of Computing, 1*(4), 349-368. https://doi.org/10.3934/mfc.2018017

Tu, B., Wang, F., Huo, Y., & Wang, X. (2023). A hybrid algorithm of grey wolf optimizer and Harris hawks optimization for solving global optimization problems with improved convergence performance. *Scientific Reports, 13*(1), 22909. https://doi.org/10.1038/s41598-023-49754-2

Tzanetos, A., & Blondin, M. (2023). A qualitative systematic review of metaheuristics applied to tension/compression spring design problem: current situation, recommendations, and research direction. *Engineering Applications of Artificial Intelligence, 118*, 105521. https://doi.org/10.1016/j.engappai.2022.105521

Utama, D.M., Fitriani, U., Amallynda, I., & Azmi, R.D. (2022). A novel hybrid yellow saddle goatfish algorithm for fuel consumption vehicle routing problem with simultaneous pick-up and delivery problem. *Jurnal Teknik Industri, 23*(1), 43-66. https://doi.org/10.9744/jti.23.1.43-66

Varshney, M., Kumar, P., Ali, M., & Gulzar, Y. (2024). Dynamic random walk and dynamic opposition learning for improving Aquila optimizer: solving constrained engineering design problems. *Biomimetics, 9*(4), 215. https://doi.org/10.3390/biomimetics9040215

Zaldivar, D., Morales, B., Rodriguez, A., Valdivia-G., A., Cuevas, E., & Perez-Cisneros, M. (2018). A novel bio-inspired optimization model based on yellow saddle goatfish behavior. *Biosystems, 174*, 1-21. https://doi.org/10.1016/j.biosystems.2018.09.007

Zamani, H., Nadimi-Shahraki, M.H., Mirjalili, S., Gharehchopogh, F.S., & Oliva, D. (2024). A critical review of moth-flame optimization algorithm and its variants: structural reviewing, performance evaluation, and statistical analysis. *Archives of Computational Methods in Engineering, 31*(4), 2177-2225. https://doi.org/10.1007/s11831-023-10037-8

Zhang, Y., & Cai, Y. (2024). Adaptive dynamic self-learning grey wolf optimization algorithm for solving global optimization problems and engineering problems. *Mathematical Biosciences and Engineering, 21*(3), 3910-3943. https://doi.org/10.3934/mbe.2024174

Zhang, Y., Jin, Z., & Chen, Y. (2020). Hybrid teaching–learning-based optimization and neural network algorithm for engineering design optimization problems. *Knowledge-Based Systems, 187*, 104836. https://doi.org/10.1016/j.knosys.2019.07.007

Zheng, Y.-J. (2015). Water wave optimization: a new nature-inspired metaheuristic. *Computers & Operations Research, 55*, 1-11. https://doi.org/10.1016/j.cor.2014.10.008