Marine Predators Algorithm: A Nature-inspired Metaheuristic

Afshin Faramarzi [a,*1], Mohammad Heidarinejad [a], Seyedali Mirjalili [b], Amir H. Gandomi[c]

[a] Department of Civil, Architectural, and Environmental Engineering, Illinois Institute of Technology, Chicago, IL, USA

[b] Institute for Integrated and Intelligent Systems, Griffith University, Nathan, Brisbane, QLD 4111, Australia

[c] Faculty of Engineering & Information Technology, University of Technology Sydney, Australia

Abstract

This paper presents a nature-inspired metaheuristic called Marine Predators Algorithm (MPA) and its application in engineering. The main inspiration of MPA is the widespread foraging strategy namely Lévy and Brownian movements in ocean predators along with optimal encounter rate policy in biological interaction between predator and prey. MPA follows the rules that naturally govern in optimal foraging strategy and encounters rate policy between predator and prey in marine ecosystems. This paper evaluates the MPA's performance on twenty-nine test functions, test suite of CEC-BC-2017, three engineering benchmarks, and two real-world engineering design problems in the areas of ventilation and building energy performance. MPA is compared against three classes of existing optimization methods, including 1) GA and PSO as the most well-studied metaheuristics, 2) GSA, CS and SSA as almost recently developed algorithms and 3) CMA-ES, SHADE and LSHADE-cnEpSin as high performance optimizers and winners of IEEE CEC competition. Among all methods, MPA gained the second rank and demonstrated very competitive results compared to LSHADE-cnEpSin as the best performing method and one of the winners of CEC 2017 competition. The statistical post hoc analysis revealed that MPA can be nominated as a high-performance optimizer and is a significantly superior algorithm than GA, PSO, GSA, CS, SSA and CMA-ES while its performance is statistically similar to SHADE and LSHADE-cnEpSin.

Keywords

Marine Predators Algorithm; Metaheuristic; Stochastic Optimization; Global Optimization; Evolutionary Computation; Swarm Intelligence

1. Introduction

Based on the "*survival of the fittest*" theory, predators must choose an optimal strategy to maximize their encounter rates with prey in natural environments (Viswanathan et al., 1999). In general, the foraging pattern of many animals in nature is effectively a random walk strategy; a stochastic process in which the next state/position is dependent on the current state and a transition probability to the next location which can be mathematically modeled (Frederic Bartumeus, da Luz, Viswanathan, & Catalan, 2005). These optimal strategies have been evolved by the ecosystem and naturally picked by predators to survive.

[1] Corresponding Author
Email: afaramar@hawk.iit.edu

One special class of random walks is called Lévy flight/walk, which is based on an optimal search theory (Humphries et al., 2010). The family of such search strategies describes a movement, characterized by many small steps associated with longer relocations which are drawn from a probability distribution with a power-law tail. It is theorized that Lévy flight/walk to be the most efficient search pattern for patchy prey in low concentrations natural environment (Humphries et al., 2010). There are studies that show many animals and marine creatures follow a Lévy flight pattern as their optimal foraging policy (Humphries et al., 2010; Reynolds & Frye, 2007; Sims et al., 2008; Viswanathan et al., 1996).

Among marine creatures, many species including sharks, tunas, marlines, sunfish and swordfish exhibits Lévy-like behavior in searching for prey (Humphries et al., 2010). Fundamental researches have collected marine predators' behavior data to demonstrate that Lévy strategy evolved as an optimal search policy among predators in response to patchy prey distribution (Humphries et al., 2010; Sims et al., 2008; Viswanathan, Raposo, & da Luz, 2008). Other studies simulated the effect of size and velocity ratio of predator to prey to find out in which ratio the maximum encounter rate between predator and prey will occur (F. Bartumeus, Catalan, Fulco, Lyra, & Viswanathan, 2002). They showed that the velocity ratio of prey to predator significantly impacts different strategies. The tradeoff between the Lévy strategy and Brownian provides an opportunity to find the optimal strategy for an optimization method that is the main inspiration of the current study. Therefore, this section summarizes the main differences and similarities between these two methods and specifies opportunities to find the best optimal strategy.

Error! Reference source not found. illustrates an example of influential variables on the decision-making in Brownian and Levy strategies (F. Bartumeus et al., 2002). In this figure, v is the velocity ratio of prey to predator, and L is the size interval of a system in which the simulation is carried out. The parameter γ indicates the ratio between the encounter rates for the Lévy and Brownian predators, and $\gamma > 1$ presents an advantage for the predator adopting a Lévy strategy. This figure shows that in small ratio velocities $(v = 0.1)$ either prey is moving in Brownian or Lévy, the best strategy for a predator is to have Lévy steps. In the unit velocity ratio $(v = 1)$, if prey moves in Lévy, predator should move in Brownian with independency of system size. In high-velocity scenarios $(v \geq 10)$, again, either prey is moving in Brownian or Lévy the best strategy for a predator is not choosing Lévy and in its optimal conditions is not moving at all, because prey will come by itself (F. Bartumeus et al., 2002). Overall, the visualizations in**Error! Reference source not found.** indicate that an optimal strategy for searching and foraging requires considering a combination of Lévy and Brownian strategies.

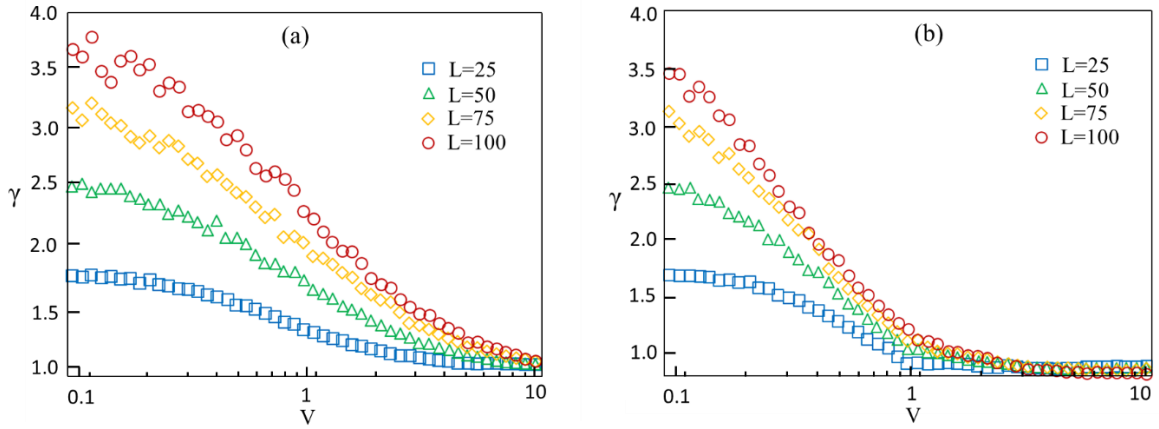


Figure 1. γ vs. v for (a) Brownian prey and (b) Lévy prey (data extracted from (F. Bartumeus et al., 2002) to reproduce this figure)

Humphries et al. (Humphries et al., 2010) show that Lévy strategy is a widespread pattern among marine predators when searching for the food in a prey-sparse environment, but when it comes to foraging in a prey-abundant area, the pattern is prevalently switched to Brownian type. Their research includes the frequency of Lévy/Brownian behavior of open-ocean predators in frontal/shelf habitats as productive area and off-shelf as a less productive environment in the northeast Atlantic and Central eastern Pacific. Figure 2 shows the frequency of behavior type in which verifies the Lévy flight foraging (LFF) hypothesis (Frederic Bartumeus, 2007).

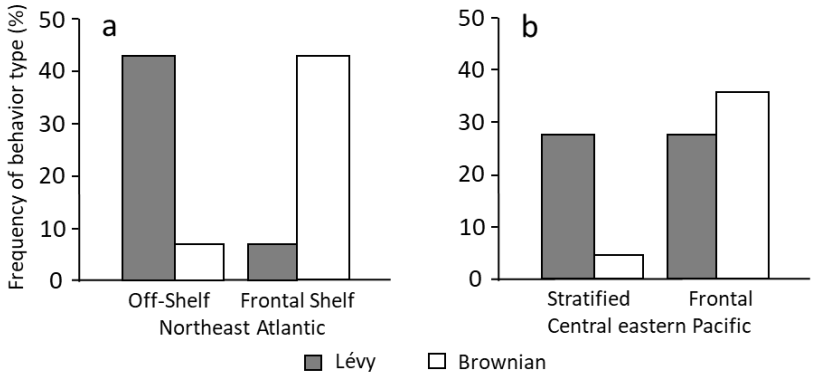


Figure 2. Spatial occurrence of Lévy and Brownian behavior of marine predators in the (a) Northeast Atlantic and (b) Pacific oceans (data extracted from (Humphries et al., 2010) to reproduce the figure)

Figure 2 is the outcome of recorded data of couple marine predator's movements such as Sharks, tunas, billfish and ocean sunfish in the mentioned oceans over 5,700 days, including 12,294,347 steps. Figure 2(a) shows that the behavior of a predator in off-shelf of the Atlantic Ocean is 80% Lévy and 20% Brownian, but in the frontal shelf, it is precisely opposite with 20% Lévy and 80% Brownian. During the lifetime of a predator which travels between off-shelf and frontal shelf and according to this figure, it is possible to conclude that the total behavior of Lévy and Brownian is exactly 50% on each. Figure 2(b) illustrates the total behavior of a predator in the Pacific Ocean is about 44% Brownian and 56% Lévy which can be considered as 50%. Therefore, the conclusion is that a marine predator spends the whole of its life showing almost 50% of Brownian motion and 50% of Lévy movement traversing different habitats in the ocean. This research also reports an

anomalous behavior of few predators in the Central Eastern Pacific moving in the convergence zone stratified water. Although this zone belongs to the prey-abundant environment, predators in this area exhibit Lévy-like movement with longer vertical jumps. The reason behind this issue is associated with the formation of a mesoscale eddy that is common in these regions (Zainuddin, Kiyofuji, Saitoh, & Saitoh, 2006). This kind of long jumps is usually in an effort to find another spot of patchily distributed prey.

Another similar behavior of silky sharks around drifting Fish Aggregating Device (FAD) (which are devices to attract ocean fishes for different purposes) is recorded in Western Indian Ocean where ten silky sharks are equipped with pressure sensors tags for behavioral studies (Filmalter, Dagorn, Cowley, & Taquet, 2011). It is observed that most sharks spent more than 80% of their time within 35 m of the surface in the immediate proximity of the FAD associated with the sudden and long vertical movement. These vertical jumps intrinsically are different from those in Lévy behavior. In Lévy motion, a predator freely searches the environment without any barrier, while encountering this kind of environmental issues (eddy formation and FADs) they are somehow forced to take these longer jumps. Figure 3 shows a recorded vertical movement of two silky sharks around a FAD in the Indian Ocean. The longer jumps are specified by the red oval. This observation confirms that these directional changes are most likely occurred to hopefully find a prey-abundant environment.

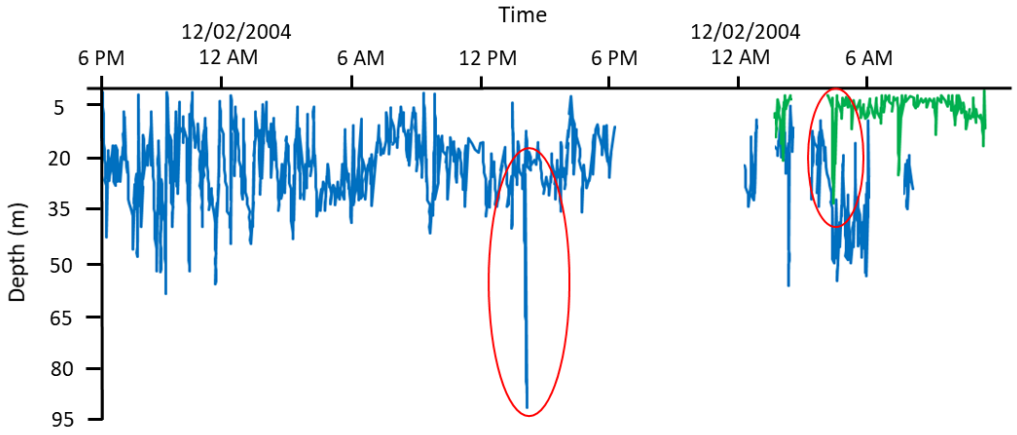


Figure 3. Detailed vertical movement of two silky sharks (blue and green lines) around FAD in the Indian ocean (data extracted from (Filmalter et al., 2011) *to reproduce the figure)*

There are also some studies which demonstrated that marine predators benefit from cognitive skills and spatial memory helping them in activities like food retrieval, mate choice, and habitat selection (Clark, 1959; Dugatkin & Wilson, 1992; Schluessel & Bleckmann, 2012). Clark (Clark, 1959) reported that Lemon sharks similar to other fishes are talented to quickly learn to return to the place where they are usually fed. This memory can also help them on how to obtain food which lasts up to ten weeks.

The following highlights summarize the governing policies for the optimal foraging and interactions and memories in marine predators:

- Marine predators use Lévy strategy for the environment with a low concentration of prey while employing Brownian movement for the areas with abundant prey;
- They showed the same percentages of Lévy and Brownian movement during their lifetime of traversing different habitats;
- Due to the environmental effects such as natural (eddy formation) or human-caused (FADs), they change their behavior to hopefully find areas with a different distribution of prey;
- In low-velocity ratio ($v = 0.1$), the best strategy for a predator is Lévy; either prey is moving in Brownian or Lévy;
- In the unit velocity ratio ($v = 1$), if prey moves in Lévy, the best strategy for a predator is Brownian. Other scenarios are dependent on system size;
- In high-velocity ratio ($v \geq 10$) the best strategy for a predator is not moving at all. In this case, either prey is moving Brownian or Lévy; and
- They take advantage of good memory in reminding of their associates as well as the location of successful foraging.

Overall, based on the extracted rules and points governing the foraging policy of marine predators, this study establishes a novel optimization algorithm called Marine Predator Algorithm (MPA). Section 2 discusses related studies. MPA is proposed in Section 3. The results on test functions are presented and analyzed in Section 4. Sections 5 and 6 summarize the results of engineering design and real-world problems. Finally, Section 7 concludes the work and suggests a future direction.

2. Literature review

Optimization methods generally fall in two classes of deterministic and stochastic. Deterministic methods of optimization are either gradient-based or non-gradient based. The type that use gradient information to find global solution are mathematical programming methods including linear and non-linear programming (Boyd & Vandenberghe, 2004; Faramarzi & Afshar, 2012, 2014), while the other type of deterministic approaches use conditions other than the gradient information to find global solution (Lera & Sergeyev, 2018; Liuzzi, Lucidi, & Piccialli, 2010; Yaroslav D. Sergeyev & Kvasov, 2006). One inherent drawback of mathematical programming methods is a high probability of stagnation in local optima during the search of non-linear space. To overcome the issue, one might try a different initial design, modify the methods and hybridize them with different algorithms, which sometimes make them unique for that specific problem (Faramarzi & Afshar, 2014). One of the drawbacks of non-gradient deterministic approaches is that their implementation is not easy and requires high mathematical preparation to understand and use (Ya D. Sergeyev, Kvasov, & Mukhametzhanov, 2018).

Another alternative to these issues is applying metaheuristic algorithms that fall under the category of stochastic optimization. Metaheuristics employ random operators and variables to globally search the space while avoid trapping in local optima. Their understanding and implementation are quite easier than the other methods. Although these methods do not guarantee to obtain the global solution at the last iteration, there are some characteristics which makes them very popular among all class of optimization methods. The near-global solutions, independence to the problem,

flexibility, and gradient-free nature of these methods are highlighted features that can be accounted as the reason for their widespread use (Mirjalili, Mirjalili, & Lewis, 2014).

Based on the inspiration source, these methods are mainly categorized into three classes of (i) evolutionary algorithms, (ii) swarm intelligence, and (iii) physics-based methods.

Evolutionary algorithms use the concept of biological evolution which naturally happens in nature. The most well-known method in this class is Genetic Algorithm (GA) (Holland, 1975) which uses two concepts of cross-over and mutation to search the domain and improve the results from an initial random population. There are other well-known evolutionary algorithms such as Evolution strategy (ES) (Hansen, Müller, & Koumoutsakos, 2003), Genetic programming (GP) (Koza, 1992), and Differential Evolution (DE) (Storn & Price, 1997).

Swarm intelligence is another class of optimization method. These techniques mimic the intelligent and collective behavior of swarms, herds, flocks, and schools of different creatures in nature. Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995) is the most well-known method in this class. which is inspired by the collective behavior of birds and fish schools. Other methods in the literature are Ant Colony Optimization (ACO) (Dorigo, Birattari, & Stutzle, 2006), Grey Wolf Optimizer (Mirjalili, Saremi, Mirjalili, & Coelho, 2016), Krill Herd (KH) (Gandomi & Alavi, 2012), Cuckoo Search (CS) (X.-S. Yang & Deb, 2009), Salp Swarm Algorithm (SSA) (Mirjalili et al., 2017).

Physics-based methods were inspired by the different physical laws in nature. The interaction of search agents in these methods is usually governed by rules and laws extracted from physical phenomena. Perhaps, the most well-known algorithm in this class is Simulated Annealing (SA) (Kirkpatrick, Gelatt, & Vecchi, 1983). This method employs thermodynamic laws of heating up a material and then slowly cooling down. Another well-known method in this category is Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-pour, & Saryazdi, 2009) which uses Newtonian law of gravity and interaction between masses to update the position of search agents toward the optimum points. Due to the popularity of these methods, there are some books that have gathered different methods of each category along with literature review and examples on real-world applications from different engineering disciplines (Mirjalili, Dong, & Lewis, 2020; X.-S. Yang, Cui, Xiao, Gandomi, & karamanoglu, 2013).

3. Background

Before introducing the steps of the proposed algorithm, there is a need to know a mathematical model for two main random walks of (i) Brownian and (ii) Lévy motion.

3.1 Brownian motion

The standard Brownian motion is a stochastic process in which their step length is drawn from the probability function defined by Normal (Gaussian) distribution with zero mean ($\mu = 0$) and unit variance ($\sigma^2 = 1$). The governing Probably Density Function (PDF) at point x for this motion is as follows (Einstein, 1956):

$$f_B(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right) \quad (1)$$

3.2 Lévy flight

Lévy flight is a type of random walk which the step sizes are determined from a probability function defined by Lévy distribution (power-law tail):

$$L(x_j) \approx |x_j|^{1-\alpha} \quad (2)$$

where x_j is the flight length, and $1 < \alpha \leq 2$ is the power-law exponent (Humphries et al., 2010). The probability density of Lévy stable process in integral form is defined as (Mantegna, 1994):

$$f_L(x;\alpha,\gamma) = \frac{1}{\pi}\int_0^{\infty} \exp(-\gamma q^{\alpha})\cos(qx)\,dq \quad (3)$$

Where α defines the distribution index and controls the scale properties of the process while γ selects the scale unit. The integral in Eq. (3) has an analytical solution in just a few cases. When α is equal to 2, it represents a Gaussian distribution, and when α is equal to 1, it shows a Cauchy distribution (X. Yang, 2010). The solution for the integral in Eq. (3) generally requires using the series expansion method only when x owns a huge value as follows:

$$f_L(x;\alpha,\gamma) \approx \frac{\gamma\Gamma(1+\alpha)\sin\left(\frac{\pi\alpha}{2}\right)}{\pi x^{(1+\alpha)}} \quad , \quad x \to \infty \quad (4)$$

Where Γ stands for Gamma function in which for integer α numbers, $\Gamma(1+\alpha)$ is equal to $\alpha!$

Mantegna (Mantegna, 1994) proposed an accurate and fast algorithm for generating a Lévy stable process for an arbitrary value of index distribution (α) ranged in 0.3 and 1.99. This study uses the Magneta method for generating random numbers based on Lévy distribution as follow:

$$Levy(\alpha) = 0.05 \times \frac{x}{|y|^{1/\alpha}} \quad (5)$$

Where x and y are two normal distribution variables with standard deviations of σ_x and σ_y as follows:

$$x = Normal(0,\sigma_x^2) \quad (6)$$

$$y = Normal(0,\sigma_y^2) \quad (7)$$

In Eq. 6, σ_x is calculated as follows:

$$\sigma_x = \left[\frac{\Gamma(1+\alpha)\sin(\frac{\pi\alpha}{2})}{\Gamma\left(\frac{(1+\alpha)}{2}\right)\alpha\, 2^{\frac{(\alpha-1)}{2}}}\right]^{1/\alpha} \quad and \;\; \sigma_y = 1 \;\; and \;\; \alpha = 1.5 \quad (8)$$

Figure 4 depicts Lévy and Brownian distribution along with their trajectories in two- and three-dimensional spaces. Figure 4(a) and Figure 4(d) show the distribution of Lévy and Brownian motion in a 1-dimensional perspective. Figure 4(b) and Figure 4(e) illustrate 2D trajectories for the Lévy and Brownian, respectively, drawn from their distributions. Finally, Figure 4(c) and Figure 4(f) illustrate the 3D version of these trajectories. As the related figures show, walks from Lévy strategy trace the domain mostly with small steps associated with long jumps while the Brownian motion can cover areas of the domain with more uniform and controlled steps compared to Lévy strategy.

Based on the characteristics shown in Figure 4, neither the Lévy flight nor Brownian motion is solely effective enough to globally and locally search a domain by themselves. However, their combination and proper use of each strategy can provide a systematic explorer-exploiter framework which can work more efficient than each strategy by itself. As it is seen in the Figure, Lévy flight is mostly associated with tiny steps and occasion long jumps. This characteristic can be employed as improved searchability in optimization literature which shows more efficient performance compared to uniform random search (X.-S. Yang & Deb, 2013). The figure demonstrates while the Lévy flight is able to efficiently and deeply search a nearby neighborhood due to small step lengths and explore other areas of the domain due to its long steps, it cannot solely cover all areas of a domain. On the other hand, the walks from Brownian motion, which is clearly observable in the figure, can trace and explore distant areas of the neighborhood but cannot search as accurate and deep as Lévy strategy. One can notice that taking advantage of the unique characteristics of each random walk and combining these strategies together how a domain can be more globally and locally explored and exploited which can be inferred from Figure 4 (b) and Figure 4 (e). Thus, this study is trying to present an efficient optimization method called Marine Predators Algorithm (MPA) benefiting from the unique characteristics of Lévy strategy along with features of Brownian motion which is proven to be more efficient in exploration and exploitation of a domain in optimization literature.

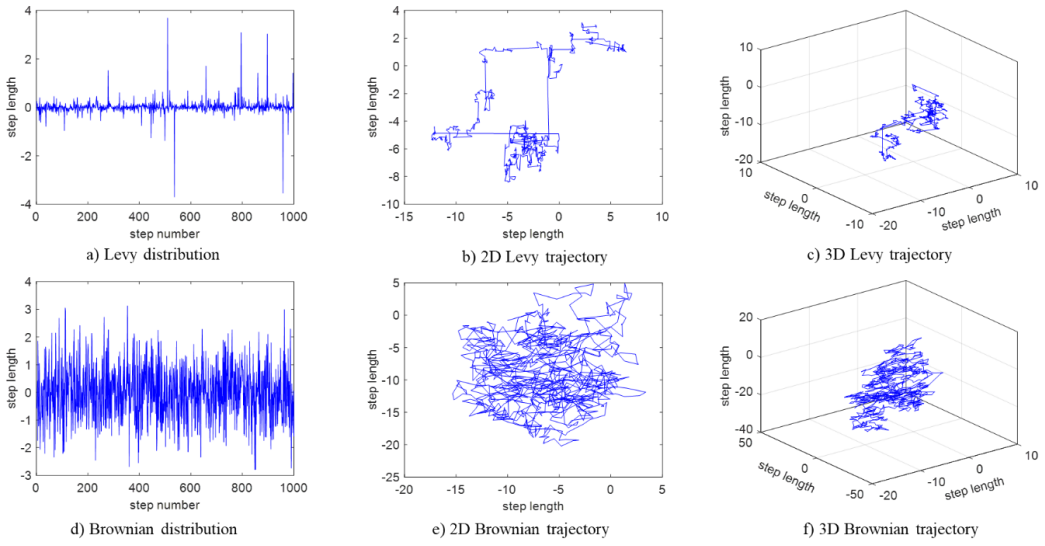


Figure 4. Lévy flight (a) distribution, (b) 2D trajectory, (c) 3D trajectory vs. Brownian motion (d) distribution, (e) 2D trajectory, and (f) 3D trajectory

4. Marine Predators Algorithm

This section explores the development process of the MPA algorithm as a simple and efficient metaheuristic optimization method.

4.1 MPA Formulation

Similar to most of the metaheuristics, MPA is a population-based method, in which the initial solution is uniformly distributed over the search space as the first trial:

$$X_0 = X_{min} + rand\,(X_{max} - X_{min}) \tag{9}$$

Where X_{min} and X_{max} are the lower and upper bound for variables and *rand* is a uniform random vector in the range of 0 to 1.

Based on the *survival of the fittest theory*, it is said that top predators in nature are more talented in foraging. Thus, the fittest solution is nominated as a top predator to construct a matrix which is called Elite. Arrays of this matrix oversee searching and finding the prey based on the information on prey's positions.

$$Elite = \begin{bmatrix} X_{1,1}^{I} & X_{1,2}^{I} & \cdots & X_{1,d}^{I} \\ X_{2,1}^{I} & X_{2,2}^{I} & \cdots & X_{2,d}^{I} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ X_{n,1}^{I} & X_{n,2}^{I} & \cdots & X_{n,d}^{I} \end{bmatrix}_{n\times d} \tag{10}$$

Where $\overrightarrow{X^I}$ represents the top predator vector, which is replicated n times to construct the *Elite* matrix. n is the number of search agents while d is the number of dimensions. It is noted that both predator and prey are considered as search agents. Because by the time that a predator is looking for its prey, the prey is looking for its own food. At the end of each iteration, the *Elite* will be updated if the top predator is substituted by the better predator.

Another matrix with the same dimension as *Elite* is called *Prey* which the predators update their positions based on it. In a simple word, the initialization creates the initial *Prey* of which the fittest one (predator) constructs the *Elite*. The *Prey* is shown as follows:

$$Prey = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,d} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,d} \\ X_{3,1} & X_{3,2} & \cdots & X_{3,d} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,d} \end{bmatrix}_{n\times d} \tag{11}$$

In Eq. (11), $X_{i,j}$ presents the *j-th* dimension of *i-th* prey. It should be noted that the whole process of the optimization is mainly and directly related to these two matrices.

4.2 MPA Optimization Scenarios

MPA optimization process is divided into three main phases of optimization considering different velocity ratio and at the same time mimicking the entire life of a predator and prey: (1) in high velocity ratio or when prey is moving faster than predator, (2) in unit velocity ratio or when both predator and prey are moving at almost same pace, and (3) in low velocity ratio when predator is moving faster than prey. For each defined phase, a specific period of iteration is specified and assigned. These steps are defined based on the rules governed on the nature of predator and prey movement while mimicking the movement of predator and prey in nature. These three phases include:

Phase 1: In high-velocity ratio or when predator is moving faster than prey. This scenario happens in the initial iterations of optimization, where the exploration matters. Based on the rules extracted from Fig. 1, in high-velocity ratio ($v \geq 10$), the best strategy for predator is not moving at all. The mathematical model of this rule is applied as:

While $Iter < \frac{1}{3} Max\_Iter$

$$\begin{aligned} \overrightarrow{stepsize_i} &= \vec{R}_B \otimes (\overrightarrow{Elite_i} - \vec{R}_B \otimes \overrightarrow{Prey_i}) \qquad i = 1, \dots n \\ \overrightarrow{Prey_i} &= \overrightarrow{Prey_i} + P.\vec{R} \otimes \overrightarrow{stepsize_i} \end{aligned} \tag{12}$$

where R_B is a vector containing random numbers based on Normal distribution representing the Brownian motion. The notation $\otimes$ shows entry-wise multiplications. The multiplication of R_B by prey simulates the movement of prey. P=0.5 is a constant number, and R is a vector of uniform random numbers in [0,1]. This scenario happens in the first third of iterations when the step size or the velocity of movement is high for high exploration ability. Iter is the current iteration while Max_iter is the maximum one.

Phase 2: In unit velocity ratio or when both predator and prey are moving at the same pace. It mimics that both of them are looking for their prey. This section occurs in the intermediate phase of optimization where the exploration tries to be transiently converted to exploitation. In this phase, both exploration and exploitation matters. Consequently, half of the population is designated for exploration and the other half for exploitations. In this phase, prey is responsible for exploitation and predator for exploration. Based on the rule, in the unit velocity ratio ($v \approx 1$), if prey moves in Lévy, the best strategy for predator is Brownian. Thus, this study considers prey moves in Lévy while predator moves in Brownian.

While $\frac{1}{3} Max\_Iter < Iter < \frac{2}{3} Max\_Iter$

For the first half of the population

$$\begin{aligned} \overrightarrow{stepsize_i} &= \vec{R}_L \otimes (\overrightarrow{Elite_i} - \vec{R}_L \otimes \overrightarrow{Prey_i}) \quad i = 1, .., n/2 \\ \overrightarrow{Prey_i} &= \overrightarrow{Prey_i} + P.\vec{R} \otimes \overrightarrow{stepsize_i} \end{aligned} \tag{13}$$

where $\vec{R}_L$ is a vector of random numbers based on Lévy distribution representing Lévy movement. The multiplication of $\vec{R}_L$ and *Prey* simulates the movement of prey in Lévy manner while adding

the step size to prey position simulates the movement of prey. Since most of the Levy distribution step size is associated with small steps, this section is helping to exploitation. For the second half of the populations, this study assumes:

$$\overrightarrow{stepsize_i} = \vec{R}_B \otimes (\vec{R}_B \otimes \overrightarrow{Elite_i} - \overrightarrow{Prey_i}) \quad i = n/2, .., n$$
$$\overrightarrow{Prey_i} = \overrightarrow{Elite_i} + P.CF \otimes \overrightarrow{stepsize_i} \tag{14}$$

While CF=$(1 - \frac{Iter}{Max\_Iter})^{(2\frac{Iter}{Max\_Iter})}$ is considered as an adaptive parameter to control the step size for predator movement. Multiplication of $\vec{R}_B$ and *Elite* simulates the movement of predator in Brownian manner while prey updates its position based on the movement of predators in Brownian motion.

phase 3: In low-velocity ratio or when predator is moving faster than prey. This scenario happens in the last phase of the optimization process which is mostly associated with high exploitation capability. In low-velocity ratio $(v = 0.1)$ the best strategy for predator is Lévy. This phase is presented as:

While $Iter > \frac{2}{3} Max\_Iter$

$$\overrightarrow{stepsize_i} = \vec{R}_L \otimes (\vec{R}_L \otimes \overrightarrow{Elite_i} - \overrightarrow{Prey_i}) \quad i = 1, .., n$$
$$\overrightarrow{Prey_i} = \overrightarrow{Elite_i} + P.CF \otimes \overrightarrow{stepsize_i} \tag{15}$$

Multiplication of $\vec{R}_L$ and *Elite* simulates the movement of predator in Lévy strategy while adding the step size to *Elite* position simulates the movement of predator to help the update of prey position.

This study simulates the movement of predators and prey according to what happens in nature based on the rules and points extracted from different literature. These phases simulate the step size taken by a predator to catch prey. Based on the rules, it is reasonable to assume the same percentage of Lévy and Brownian movement during the lifetime of a predator. In the first phase, predator is not moving at all while in the second phase it is moving in Brownian and finally in the third phase, it shows Lévy strategy. This scenario also happens to prey because the prey is another potential predator, e.g. silky sharks and tuna fish. Both of them are considered as marine predators, but while tuna fish is prey for a silky shark, it is a predator for bony fishes and marine invertebrates. In the first phase, prey is moving in Brownian and in the second phase, it follows Lévy behavior. The strategy which one-third of iterations allocated to each phase is experimentally achieved to be optimized and give slightly better results compared to the strategies which switch between these stages or cyclically repeat of the stage. Since this is the first version of the method, interested readers can improve this method by defining other criteria on how and when the algorithm uses each phase for an update.

4.3 Eddy formation and FADs' effect

Another point which causes a behavioral change in marine predators is environmental issues such as the eddy formation or Fish Aggregating Devices (FADs) effects. Based on the study of Filmalter

et al. (Filmalter et al., 2011), sharks spend more than 80% of their time in the immediate vicinity of FADs, and for the rest 20%, they will take a longer jump in different dimensions probably to find an environment with another prey distribution. The FADs are considered as local optima and their effect as trapping in these points in search space. Consideration of these longer jumps during simulation avoids stagnation in local optima. Thus, the FADs effect is mathematically presented as:

$$\overrightarrow{Prey_i} = \begin{cases} \overrightarrow{Prey_i} + CF\left[\vec{X}_{min} + \overrightarrow{R} \otimes \left(\vec{X}_{max} - \vec{X}_{min}\right)\right] \otimes \vec{U} & if\ r \leq FADs \\ \overrightarrow{Prey_i} + [FADs(1-r) + r]\left(\overrightarrow{Prey_{r1}} - \overrightarrow{Prey_{r2}}\right) & if\ r > FADs \end{cases} \quad (16)$$

Where $FADs = 0.2$ is the probability of FADs effect on the optimization process. $\vec{U}$ is the binary vector with arrays including zero and one. This is constructed by generating a random vector in [0,1] and changing its array to zero if the array is less than 0.2 and one if it is greater than 0.2. r is the uniform random number in [0,1]. $\vec{X}_{min}$ and $\vec{X}_{max}$ is the vector containing the lower and upper bounds of the dimensions. $r1$ and $r2$ subscripts denote random indexes of *prey* matrix.

4.4 Marine memory

Based on the highlighted points, marine predators have a good memory in reminding the place where they have been successful in foraging. This capability is simulated by memory saving in MPA. After updating the *Prey* and implementing FADs effect, this matrix is evaluated for fitness to update the *Elite*. The fitness of each solution of the current iteration is compared to its equivalent in prior iteration, and the current one replaces the solution if it is more fitted. This process also improves the solution quality with the lapse of iteration (Parouha & Das, 2016) and also simulates the returning of predators to the locations of the prey-abundant area with successful foraging. The pseudo-code of MPA is presented below.

Initialize search agents (Prey) populations i=1,…,n
***While** termination criteria are not met*
Calculate the fitness and construct the Elite matrix
  ***If** Iter<Max_Iter/3*
    Update prey based on Eq. 12
  ***Else if** Max_Iter/3<Iter<2*Max_Iter/3*
    For the first half of the populations (i=1,…,n/2)
    Update prey based on Eq. 13
    For the other half of the populations (i=n/2,…,n)
    Update prey based on Eq. 14
  ***Else if** Iter>2*Max_Iter/3*
    Update prey based on Eq. 15
  End (if)
    Accomplish memory saving and Elite update
    Applying FADs effect and update based on Eq. 16
    Accomplish memory saving and Elite update
*End **while***

---

5 MPA phases, exploration and exploitation

The three phases of optimization are schematically presented in Figure 5. In the first phase of optimization (shown as phase 1 in the figure), prey moves in Brownian motion. Since the preys

are uniformly distributed throughout the search domain in initial iterations and the distance between predator and prey is relatively large, Brownian motion can help preys to explore their neighborhood separately which results in a good exploration of the domain. Then, the prey with a new position is evaluated for the fitness, and the position is replaced if it is more fitted than the previous one. The fitted positions of prey can be interpreted as abundant food areas, and the saving procedure is equivalent to prey's memory to remember abundant food areas. If prey is more successful in foraging for its food, it can be considered as a predator. It means the fitness value of the prey is calculated and replace the top predator if it is better fitted. Now, it is time for predators to start foraging while the prey is still looking for their food. This is where the second phase of optimization starts.

This phase is designed to perform a good transition from exploration to exploitation. So, in order to take advantage of exploration and exploitation in this phase, both predator and prey search for their food. In this phase, half of the population is in charge of exploration and the other half is for exploitations. predator starts searching for its prey in Brownian motion while prey switches to Lévy strategy to efficiently search its close neighborhood and take a long jump if it could not find any food. The phase 2 shown in the figure schematically shows this phase of optimization where predator starts taking Brownian steps in blue trajectory to better search the domain while prey moves in Lévy strategy shown by green walks. Since the predator and prey locations become close to each other and the step length is going to be smaller than the previous phase, FADs effect along with long steps of Lévy strategy greatly helps MPA to avoid from local optima stagnation and better performance of the method.

As the optimization reaches its final stage, the algorithm needs high exploitation ability. In this phase, the predator starts switching its behavior from Brownian to Lévy strategy to more efficiently search a certain neighborhood. The adaptive defined convergence factor (CF) in this phase greatly helps predators to limit the search areas within a particular neighborhood for exploitation and also avoid wasting some search effort drawn from the long step sizes of Lévy strategy for non-promising regions of the domain. The last phase is depicted in the figure as phase 3.

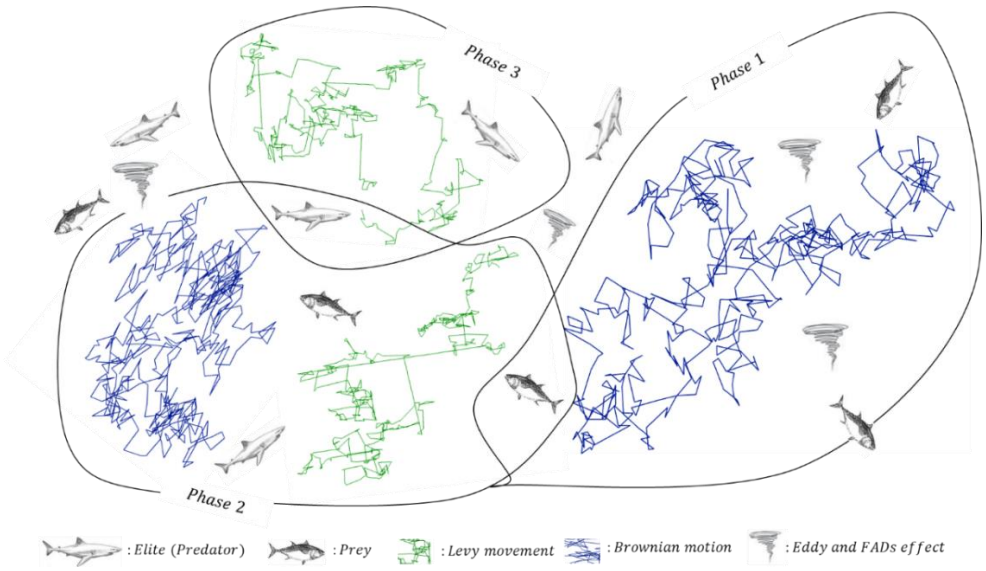


Figure 5. The three MPA optimization phases

It is noted that the computational complexity of the proposed method is of $O\left(t(nd + Cof * n)\right)$ where t is iteration, n is a number of agents, *Cof* is the cost of function evaluation and d is problem dimension.

6. Numerical experiment

This study assessed the performance of MPA using different test functions. Two classes of well-known benchmarks are employed as mathematical and engineering examples. In order to evaluate the MPA's ability to explore, exploit, and escape from local minima, the benchmark includes unimodal, multimodal, fixed dimension multimodal, and composition functions. Unimodal test functions (TF1-TF6) are designed to challenge the exploitation ability of an algorithm while multimodal functions (TF6-TF13) are used to experiment the exploration performance. These two classes of functions are tested in 50 dimensions. The fixed dimension test functions (TF14-TF23) show the exploration capability of MPA in low dimensions, and finally, composition functions (TF24-TF29) are applied to test MPA's ability for escaping from local minima. Since the complexity of composition functions are similar to real and challenging optimization problems by having too many local minima, they are designed to challenge the overall performance of an algorithm. The details of these test functions can be found in (Liang, Suganthan, & Deb, 2005). Figure 6 shows a two-dimensional representation of some mathematical functions. The mathematical models of these functions are available in (Rashedi et al., 2009).

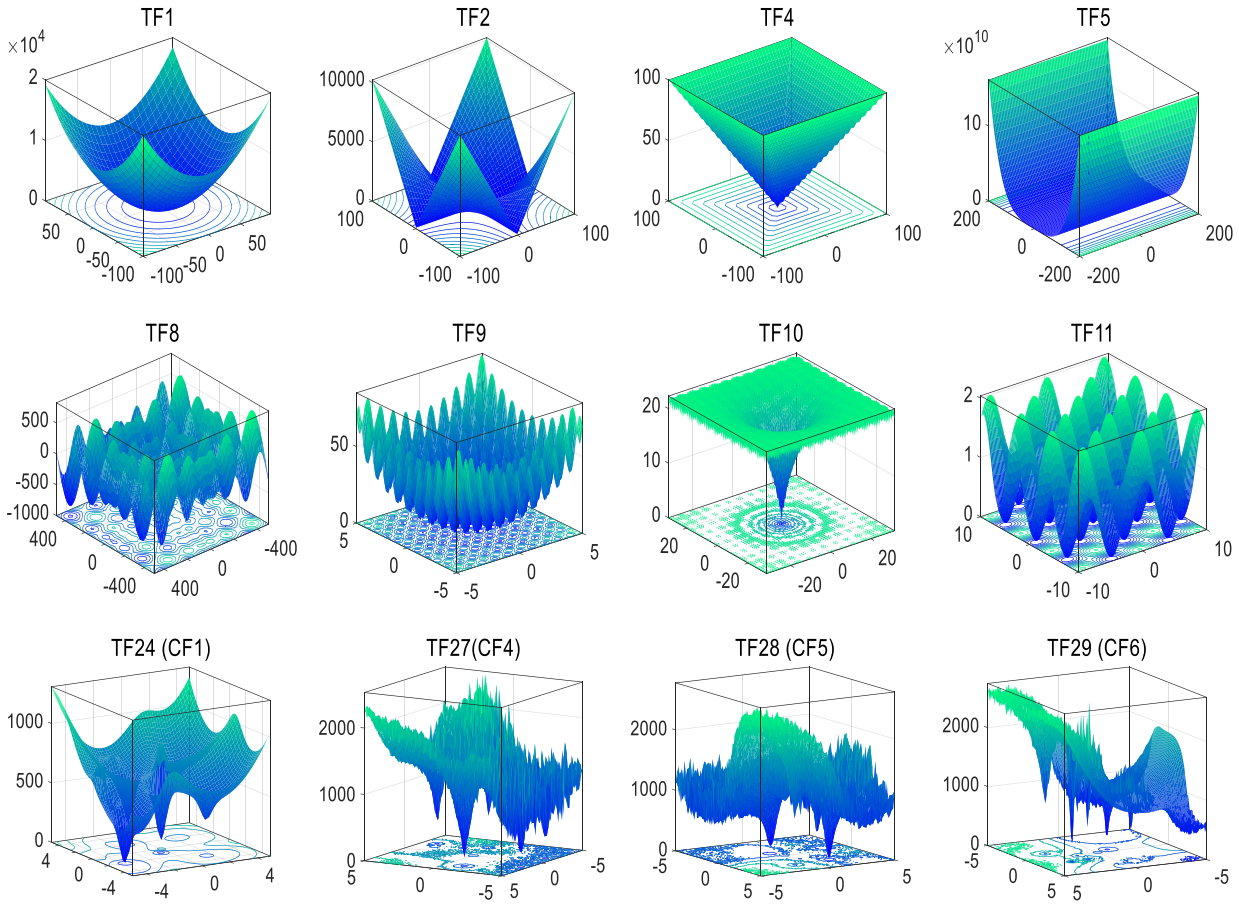


Figure 6. Two-dimensional view of some mathematical benchmark functions

500 iterations considering a maximum number of 25,000 function evaluations are used to solve the test functions using MPA and other methods. In order to have meaningful statistical results, this study ran MPA 30 times, and **Error! Reference source not found.**to**Error! Reference source not found.** provide the results, including average and standard deviation values of best-so-far solutions found in each run. To show the effectiveness superiority of MPA to that of other methods, the test is carried out for three categories of methods: i) Genetic Algorithm (GA) (Holland, 1975) and Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995) as the most well-known optimization methods ii) Gravitational Search Algorithm (GSA) (Rashedi et al., 2009), Cuckoo search (CS) (X.-S. Yang & Deb, 2009) and Salp Swarm Algorithm (SSA) (Mirjalili et al., 2017) as recently developed metaheuristics and iii) Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen et al., 2003), Success-History Based Parameter Adaptation Differential Evolution (SHADE) (Tanabe & Fukunaga, 2013) (one of the winners of CEC 2013 competition) and Ensemble Sinusoidal Differential Covariance Matrix Adaptation with Euclidean Neighborhood (Noor H. Awad, Ali, & Suganthan, 2017) (one of the winners of CEC 2013 competition) as high performance optimizers. Table 1 summarizes the parameter setting for other methods. These parameters are either exactly recommended by its developers or in the range of the recommendations to have the best performance for each algorithm (Jain, Singh, & Rani, 2019).

Table 1. parameter setting for algorithms

Algorithm	Parameter	Value
PSO	Topology	Fully connected
	Cognitive and social constant (C_1, C_2)	2, 2
	Inertia weight	Linear reduction from 0.9 to 0.1
	Velocity limit	10% of the dimension range
CS	Discovery rate (P_a)	0.25
GA	Type	Real coded
	Selection	Roulette wheel (Proportionate)
	Crossover	Whole Arithmetic Probability=0.8
	Mutation	Gaussian (Probability=0.05)
GSA	Alpha, G_0, Rnorm, Rpower	20, 100, 2, 1
SHADE	Pbest, Arc rate	0.1, 2
SSA	Leader position update probability	0.5
LSHADE-cnEpSin	H, NP_{min}, Pbest rate, Arc rate, ps,pc	5, 4, 0.11, 1.4, 0.5, 0.4

6.1 Exploitation ability of MPA

Since unimodal functions have only one global optimum, they can evaluate the exploitation ability of an algorithm. **Error! Reference source not found.**Table 2 shows the results of MPA and other methods on unimodal test functions (TF1-TF7) using the average and standard deviation values. Results indicate that MPA was able to outperform the majority of methods in almost all test functions. These results show the ability of MPA in exploitation, which can help MPA to converge towards the optimum and exploit it accurately and rapidly. This ability comes from the defined adaptive CF parameter and small steps of Lévy movements.

Table 2. Results for unimodal, multimodal and composition functions

	Function		MPA	PSO	GA	GSA	CS	SSA	CMA-ES	SHADE	LSHADE-cnEpSin
Unimodal	TF1	Ave	3.27E-21	0.0409	1.095	0.0034	210.64	0.0037	8.27E-15	1.08E-08	2.19E-04
		Std	4.61E-21	0.0416	0.4896	0.0189	81.505	0.00974	5.76E-15	1.17E-08	1.21E-04
	TF2	Ave	1.57E-12	0.0659	0.106	0.0806	15.98	5.0487	1.28E-06	0.1226	0.04134
		Std	1.42E-12	0.0864	0.0498	0.3802	4.788	2.013	2.92E-06	0.1854	0.02237
	TF3	Ave	0.0864	4236.3	25187.3	1313.88	10412.38	4343.27	9.170	265.12	70.118
		Std	0.1444	1217.9	5243.43	343.116	2456.305	2136.39	6.533	127.43	34.618
	TF4	Ave	2.6E-08	9.335	35.619	6.410	18.507	15.055	1.44E-04	1.644	3.1933
		Std	9.25E-09	1.0119	9.4072	1.535	2.463	3.195	7.00E-05	0.535	0.9194
	TF5	Ave	46.049	310.39	715.98	76.561	27288.5	434.43	52.11	60.96	59.253
		Std	0.4219	430.60	634.71	41.64	15589.9	457.70	23.15	35.535	29.012
	TF6	Ave	0.398	0.0589	0.925	2.21E-12	218.17	0.0021	5.98E-15	8.32E-09	2.94E-04
		Std	0.1914	0.1217	0.5063	5.91E-13	53.864	0.0030	5.36E-15	1.00E-08	2.71E-04
	TF7	Ave	0.0018	0.0665	0.1130	0.0926	0.4055	0.2807	0.0320	0.0294	0.00993
		Std	0.0010	0.0123	0.0355	0.0322	0.1313	0.0911	0.0077	0.0100	0.00306
Multimodal (High dimensional)	TF8	Ave	-13594.1	-10815.3	-17911.6	-3570.52	-11942.8	-12232.6	-11670.6	-14832.5	-15928.4
		Std	811.3	992.1	343.1	592.0	343.1	1063.0	884.3	418.8	516.8
	TF9	Ave	0.000	78.42	35.61	32.36	220.86	78.79	30.91	101.54	76.052
		Std	0.000	16.44	8.971	7.055	22.055	25.18	5.383	14.874	9.15
	TF10	Ave	9.69E-12	1.204	0.1844	8.94E-07	9.493	3.479	11.019	0.190	0.00363
		Std	6.13E-12	0.729	0.1487	1.54E-07	2.0936	0.8281	9.795	0.448	0.00192
	TF11	Ave	0.000	0.0128	0.6561	26.479	3.067	0.0905	2.08E-10	0.0027	0.00424
		Std	0.000	0.0130	0.1881	5.7472	0.782	0.0407	1.51E-10	0.0051	0.00638
	TF12	Ave	0.0085	0.0319	0.0344	1.0151	11.209	8.541	2.99E-13	0.0187	0.00208
		Std	0.0052	0.0560	0.0776	0.5386	7.5438	2.556	1.97E-13	0.0635	0.01135
	TF13	Ave	0.4901	0.419	0.189	10.25	1306.48	59.895	4.31E-12	0.00183	0.00711
		Std	0.1932	0.5814	0.0972	6.335	3931.79	16.745	3.44E-12	0.0041	0.00797

Multimodal (Fixed-dimensional)	TF14	Ave	0.9980	2.1825	0.9980	3.7182	0.9980	1.0311	8.1094	0.9980	0.9980
		Std	2.47E-16	2.0085	8.84E-12	2.678	5.46E-16	0.1815	5.9456	3.38E-16	0.0000
	TF15	Ave	3.07E-04	5.61E-04	2.69E-03	2.05E-03	3.97E-04	8.40E-04	1.09E-02	1.76E-03	3.07E-04
		Std	4.09E-15	4.38E-04	4.84E-03	6.64E-04	1.05E-04	2.81E-04	1.87E-02	5.06E-03	1.34E-19
	TF16	Ave	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
		Std	4.46E-16	6.64E-16	2.39E-08	2.10E-15	4.79E-16	1.48E-14	6.77E-16	0.0000	6.51E-16
	TF17	Ave	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979
		Std	9.12E-15	0.000	1.90E-06	6.14E-16	7.23E-14	5.62E-14	0.0000	1.12E-16	0.0000
	TF18	Ave	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	5.7000	3.0000	3.0000
		Std	1.95E-15	1.38E-15	2.45E-07	7.00E-14	1.85E-15	2.46E-13	1.48E+01	4.52E-16	1.92E-15
	TF19	Ave	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628
		Std	2.42E-15	2.68E-15	2.85E-08	9.33E-15	2.48E-15	3.96E-14	2.71E-15	2.71E-15	2.71E-15
	TF20	Ave	-3.3220	-3.2625	-3.2705	-3.3220	-3.3220	-3.2344	-3.2903	-3.2824	-3.3220
		Std	1.14E-11	6.05E-02	5.99E-02	1.56E-13	1.01E-07	5.87E-02	5.35E-02	5.70E-02	1.36E-15
	TF21	Ave	-10.1532	-5.3010	-6.1531	-6.5834	-10.1532	-9.9837	-6.7946	-9.7358	-10.1532
		Std	2.53E-11	2.9288	3.6282	3.6765	7.06E-07	1.3240	3.6860	1.6200	6.39E-15
	TF22	Ave	-10.4029	-7.0716	-7.9827	-10.4029	-10.4029	-9.3507	-9.2089	-10.1484	-10.1532
		Std	2.81E-11	3.6840	3.4924	3.18E-13	1.07E-04	2.569	2.7439	1.3943	6.39E-15
	TF23	Ave	-10.5364	-7.2467	-7.7652	-10.5364	-10.5364	-9.8704	-9.2797	-10.5364	-10.5364
		Std	3.89E-11	3.6582	3.7265	3.07E-13	9.60E-05	2.2936	2.8975	9.03E-15	2.47E-15
Composition	TF24	Ave	6.666	60.000	66.666	13.333	0.9580	26.666	130.000	16.666	5.90E-27
	(CF1)	Std	25.370	81.367	75.809	34.574	2.5370	52.083	172.506	37.904	2.39E-26
	TF25	Ave	15.263	157.014	108.280	183.443	32.513	29.727	166.666	31.145	26.667
	(CF2)	Std	30.442	109.456	97.953	37.658	50.392	38.582	118.418	46.813	44.976
	TF26	Ave	149.77	268.849	214.016	103.284	224.57	229.946	261.988	125.690	87.256
	(CF3)	Std	24.337	135.343	84.450	60.661	42.268	66.448	241.989	28.322	34.638
	TF27	Ave	285.424	434.281	383.928	465.947	351.204	331.424	335.733	288.199	257.605
	(CF4)	Std	38.981	131.823	110.505	153.159	30.3975	24.5213	137.596	6.8694	14.736
	TF28	Ave	2.658	179.821	47.285	231.462	13.590	27.941	177.053	33.479	3.3333
	(CF5)	Std	1.697	230.547	72.106	44.1935	8.7036	41.402	238.234	47.843	18.257
	TF29	Ave	523.721	832.199	766.151	799.730	505.055	678.896	838.685	630.822	540.232
	(CF6)	Std	90.2975	151.763	191.549	99.727	5.0723	198.244	148.136	188.898	102.759
Friedman mean rank			2.57	6.26	6.83	5.59	6.00	6.36	5.00	3.88	2.52
Rank			2	7	9	5	6	8	4	3	1

6.2 Exploration ability of MPA

Multimodal test functions have many local optima, which increase exponentially with a number of dimensions (design variables). Having more than one optimum is useful if the aim is to evaluate the exploration ability of an algorithm. TF8 through TF23 are multimodal functions in high and fixed (low) dimensions. The results of applying MPA and different algorithms on these functions are reported in Table 2**Error! Reference source not found.Error! Reference source not found.**. The table shows that MPA benefits from an outstanding exploration ability compared to other methods. MPA is able to outperform all algorithms on half of the high dimensional multimodal functions and for the other half, the results are competitive to high performance optimizers. For the fixed dimensional functions, MPA has reached to the global optimum in most problems with accuracy (Std) close to the high-performance optimizers. MPA's exploration is due to its different phases of optimization, FADs effect and Brownian motion of predators.

6.3 Local minima avoidance ability of MPA

The composition test functions (TF24-TF29) are constructed by shifting, rotating and hybridizing of some primitive unimodal and multimodal functions. These functions are designed to challenge the ability of algorithms to avoid from local optima stagnation as well as having a good ability of exploration and exploitation. Table 2**Error! Reference source not found.** shows the performance of MPA and other methods on this type of function. In all functions of this category except TF24

and TF26, the results of MPA is very competitive and sometimes better than LSHADE-cnEpSin. These results show that MPA has a well-tuned ability between exploration and exploitation while showing an excellent performance to escape from local optima. The latter characteristic is due to defined eddy and FAD's effect along with more extended relocation associated with Lévy movement.

The exploitation, exploration and local minima avoidance ability of MPA was tested and MPA proved its capability to extensively explore the domain and exploit the best solution while trying to escape from local minima stagnation. Friedman mean rank is used to specify the overall rank among the competitors. As it is clear in the last row of Table 2, MPA ranked 2nd with the very small difference compared to LSHADE-cnEpSin which ranked first. Overall, based on the results, it is reasonable to nominate both methods as joint winners which their mean rank is by far better than the other competitors, even better than SHADE and CMA-ES.

6.4 Convergence analysis of MPA

Figure 7 illustrates qualitative metrics for convergence and performance analysis of MPA in the mathematical function's testbed. The first column of the figure depicts the shape of the functions in two-dimensional view to have insight about the domain's topology. The second column of the figure represents the search history as the first metric to be discussed here. This figure shows how interactional and collective behavior between/of predator and prey helps MPA to reveal a pattern about the collective search of the agents. This pattern shows a more aggregation of agents around the optimum points in unimodal functions and more scattered behavior in multimodal and composition functions. The first characteristic of the pattern aids in exploiting the results which are desirable in unimodal functions and the latter denotes as exploring the domain which helps MPA in multimodal and composition functions search entire space.

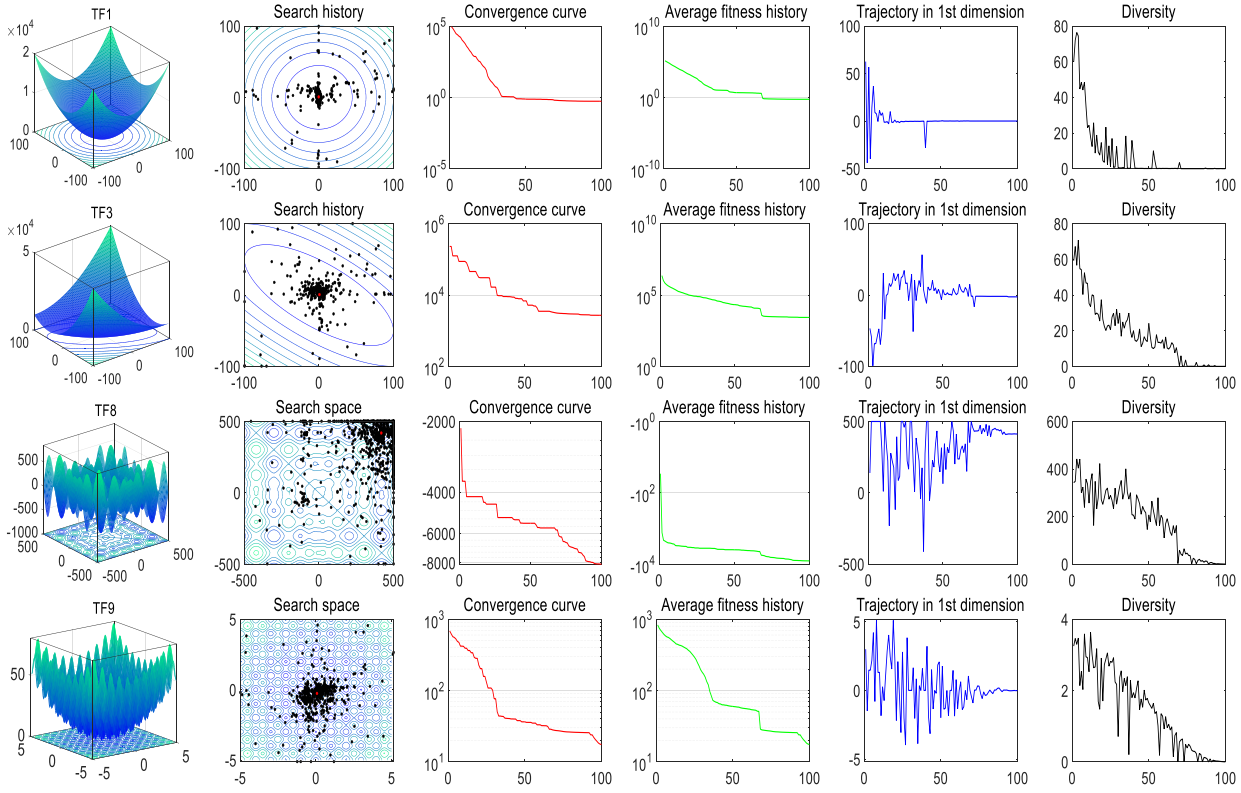
The second metric is the convergence curve which denotes as the best solution found so far. Based on each category of function, there is a specific pattern of the convergence curve shown in the figure. In unimodal functions, this pattern is relatively smooth and shows an improvement of the results with the lapse of iterations, but for multimodal and composition functions this pattern turns into a stepwise behavior which is expectable in these functions. It is possible to understand from each category that in unimodal functions MPA can recognize and encircle the optimum point at initial iterations and tries to refine the solutions as iterations pass, but in multimodal and composition functions, the agents try to globally search the domain even in the last iterations in an effort to always find better solutions. This is why no improvement of results is observed even with the passage of iterations in some multimodal functions which finally leads to the step-like pattern in these curves. It is noted that the explorative behavior owes its performance to long steps derived from the Lévy movements of the agents along with eddy and FAD's effect.

If we consider the agents (predators and preys) as members of a team, the convergence curve shows the behavior of the best player in achieving success, but it does not give any idea about the whole team's performance. This is why we used another metric to evaluate team performance in the process of optimization called average fitness history. The overall pattern of this metric is similar to the convergence curve but emphasizes how this collaborative behavior makes the results improved from the initial random population. There are some step-like behaviors observable in

average fitness history in the functions. This is due to the improvement of all agents' fitness resulting from a phase change of the algorithm which yields an overall good performance of the agents. In unimodal functions, there are mild and smooth slopes of the curve, but in multimodal and composition functions the slope trends are steep for these types.

Another metric is the trajectory of the agents shown in column 5 of the figure. This metric shows the topological variations of an agent from the beginning to the end of the optimization process. Since the agents move in many dimensions, in order to be able to have a clear track of its coordinate, we picked only the first dimension of an agent to show its trajectory. The figures related to this metric show abrupt changes having high magnitude and frequency in initial iterations which are going to be fade out in the last iterations. This trend confirms the explorative behavior in initial iterations while switching to local search in the last ones which guarantee that an algorithm can finally converge to a global/local optimum point (van den Bergh & Engelbrecht, 2006). Due to the nature of multimodal and composition functions, the magnitude and frequency of these changes are more severe than the unimodal ones and usually last longer.

The last metric is the diversity plot. This curve shows the average distance of the agents during the process. As it is seen, there is a decreasing trend in all figures presented in the last column of Figure 7 demonstrating a shift from exploration in initial phases to exploitation in the last ones. This curve tends to be more and rapidly stable in unimodal functions compared to other types.



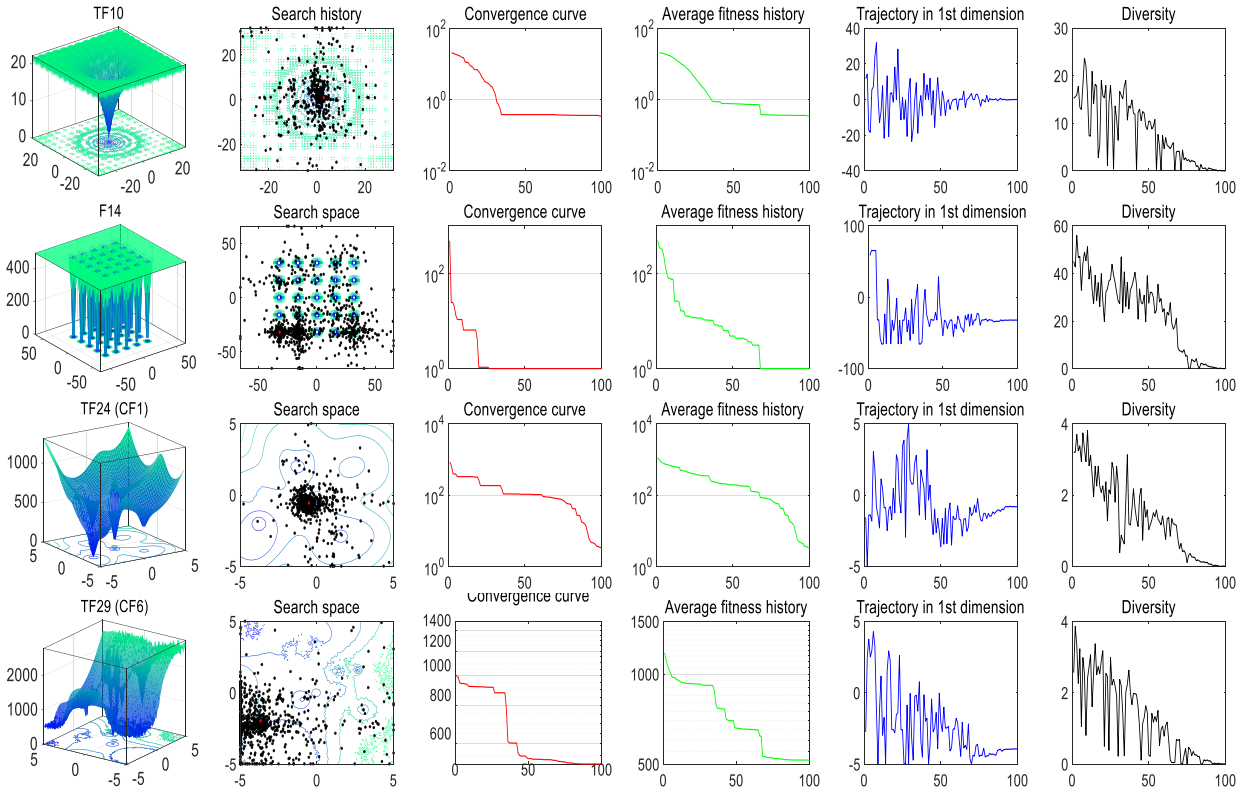


Figure 7. Search history, convergence curve, average fitness history, trajectory, and diversity

Figure 8 shows the convergence curve of MPA, PSO, GSA, CS, FA, CMA-ES and GA for some of the test functions. The results of this figure indicate a couple of distinct, recognizable behavioral patterns for MPA in optimizing the test functions. These behaviors are mainly due to different phases defined for optimization. The first behavior demonstrates a very fast convergence toward the near-global optimum point in the first phase and subsequent minor improvements in the second and third phases which verifies that just one phase could be effective to solve a problem. This behavior is observed in TF1, TF4, and TF15. Another behavior is like the previous one with no improvement of solutions in the following phases because the algorithm already reached the global or near-global point. This trend is observable in TF5, TF11. It is noted that in TF11, MPA could reach the global point in iteration 158 which is 0. Due to the semilogarithmic scale of the figure, it was impossible to show the zero in the figure, and therefore the MPA curve in TF11 is discontinued in iteration 158. The last behavior shows a convergence rate change which is subsequent to each phase change observable in TF8, TF25, TF27, and TF28. In all these figures the phase change helps MPA to explore/exploit domain by different mechanisms assigned to each phase which finally ends up in the better performance of the method. It can also be seen that the convergence curve of these figures shows a steady behavior at the final iterations of the last phase indicating the convergence guarantee. It is interesting to know, that the reason behind the definition of each phase and the effect of these phase changes in the overall performance of the algorithms is completely evident in challenging multimodal and composition functions such as TF8, TF25, TF27, and TF28. Hence, the way that MPA is designed can find the global optimum in the first phase in unimodal functions while the real performance of each defined phase is observable and effective when the algorithm is applied to challenging multimodal and composition functions.

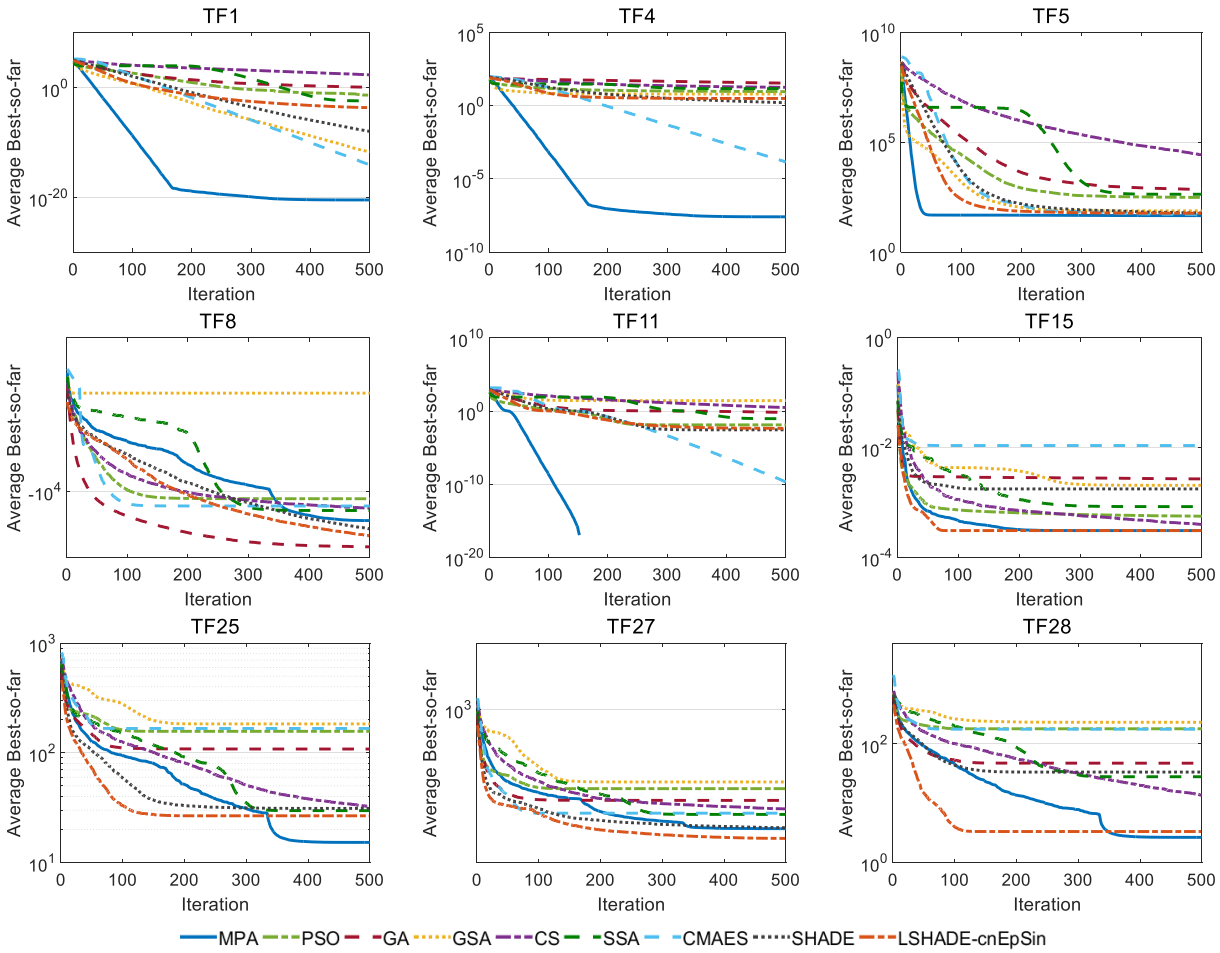


Figure 8. MPA averaged convergence curve on different mathematical test functions

6.5 Sensitivity analysis of MPA

This section focuses on sensitivity analysis of two control parameters employed in MPA. The first parameter is FADs, which controls the probability of FADs effect and its contribution to the optimization process and the second one is P which helps in attenuating/magnifying the step sizes taken by predator or prey. The analysis reveals which parameters are robust and which ones are sensitive to different inputs. MPA is able to meet a full factorial design with these two parameters on the first function from each category of unimodal (F1), multimodal (F8), fix-dimensional multimodal (F14) and composition functions (CF1). The design vector of parameters are defined as $FADs = \{0.1, 0.2, 0.5, 0.7, 0.9\}$ $and\ P = \{0.1, 0.5, 1, 1.5, 2\}$ having a totally 5*5=25 combination of design. Each design is presented as an average fitness function resulted from 25,000 function evaluations and 30 independent runs. Figure 9 shows the analysis with x-axis representing the values for control parameters while y-axis shows the average fitness as sensitivity. Figure 9(a) shows the sensitivity in TF1. As it is shown, P has more sensitive behavior than FADs in this function. FADs=0.2 and P=0.5 show optimal behavior in TF1. In TF8 which is a multimodal function, FADs shows a good performance to its left boundaries while P having this characteristic on its right boundaries. Thus, FADs=0.2 and P=2 are the best selection for this function. In F14, the parameters did not show any sensitivity. This might be associated to the simplicity of the

function or large enough number of function evaluations. For CF1 which is a composition function, FADs=0.5 and P=2 seem a good choice for optimization. Considering all types of functions, FADs=0.2 can be proposed as a conservative and initial selection when a user has no idea about the topology of their function. P=0.5 shows a very good behavior in TF1 which is unimodal function and P=2 shows this performance in multimodal and composition functions. this behavior is also aligned with the function's topology. In unimodal functions, there is a need to consider small step sizes to better exploit the results while in multimodal and composition functions these step sizes should be large enough for exploration purposes. It is observable in the figure that P shows more sensitivity in TF1 than TF8 and CF1 meaning that the range of variation of different selection of P in unimodal function is quite larger than that of in multimodal and composition functions. Thus, in order to be on the safe side, P=0.5 along with FADs=0.2 can be recommended as an initial guess of optimization.

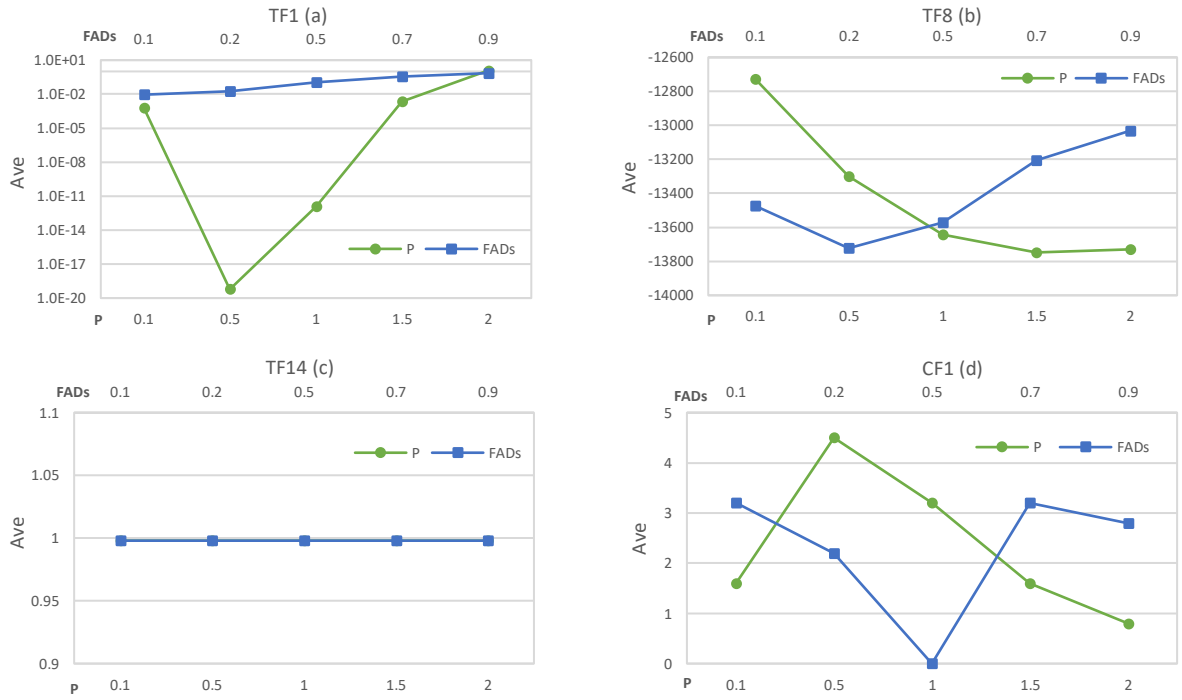


Figure 9. Sensitivity analysis of EO's control parameters with different functions

6.5 MPA performance on the CEC-BC-2017 test suite

To further demonstrate the performance of the proposed method, we selected one of the most recent and challenging test suites on the numerical optimization competitions called CEC-BC-2017 (N H Awad, Ali, Suganthan, Liang, & Qu, 2017) containing 30 functions which at least half of the functions are among the challenging hybrid and composition functions. We tested the MPA against these functions and compared the results to that of the other most well-known methods. The dimension is considered as 10 for all functions of this suite. Such as the previous section, average and standard deviation are presented in Table 3**Error! Reference source not found.**. It is noted that each method is run 30 times with 500 iterations limited to 50,000 maximum number of function evaluations. The Friedman mean rank revealed that MPA could again obtain the 2nd rank following LSHADE-cnEpSin as the best method and is followed by SHADE as the 3rd one. As it

is observable from the ranks, the performance of LSHADE-cnEpSin, MPA and SHADE are by far better than the other methods. This fact shows that MPA can be listed as a high performance optimizer.

Table 3. Results of CEC-BC-2017 test suite

	Fcn		MPA	PSO	GA	GSA	CS	SSA	CMA-ES	SHADE	LSHADE-cnEpSin
Uni modal	f1	Ave	100.00	3959.66	9799.76	296.08	100.23	3396.25	100.00	100.00	100.00
		Std	4.79E-06	4456.66	5942.54	275.24	0.135	3673.07	0.000	0.000	0.000
	f3	Ave	300.00	300.00	8721.43	10828.5	300.00	300.00	300.00	300.00	300.00
		Std	9.92E-11	1.91E-10	5900.30	1602.00	1.19E-03	9.01E-10	0.000	0.000	0.000
Multimodal	f4	Ave	400.03	405.95	410.72	406.58	400.38	406.27	400.00	400.00	400.00
		Std	0.057	3.283	18.512	2.923	0.356	10.073	0.000	0.000	0.000
	f5	Ave	510.12	513.07	516.32	556.78	520.67	521.82	530.18	503.77	502.34
		Std	3.955	6.543	6.926	8.409	4.742	10.497	58.317	1.006	0.875
	f6	Ave	600.00	600.24	600.05	621.67	602.77	609.77	682.07	600.00	600.00
		Std	6.16E-04	9.84E-01	6.69E-02	9.015	1.444	8.255	35.432	2.65E-07	2.59E-07
	f7	Ave	723.32	718.98	728.32	714.65	732.85	740.88	713.43	713.91	712.23
		Std	3.910	5.101	7.290	1.556	5.075	16.618	1.631	1.235	0.605
	f8	Ave	809.42	811.40	820.72	820.50	819.99	823.45	828.89	803.81	802.14
		Std	3.122	5.475	8.962	4.693	4.698	9.952	52.985	1.278	1.031
	f9	Ave	900.00	900.00	910.28	900.00	929.68	944.07	4667.29	900.00	900.00
		Std	1.63E-02	5.97E-14	15.154	0.000	45.37	104.65	2062.83	0.000	0.000
	f10	Ave	1437.42	1473.35	1723.35	2694.45	1699.23	1858.85	2588.01	1193.56	1070.03
		Std	141.08	214.97	252.34	297.63	142.03	294.50	414.47	84.67	56.56
Hybrid	f11	Ave	1102.93	1110.48	1125.62	1134.70	1105.73	1180.55	1111.31	1100.83	1100.02
		Std	1.27	6.95	23.80	10.48	2.04	59.80	25.44	1.36	0.12
	f12	Ave	1247.2	1.4E+04	3.7E+06	7.1E+05	1566.0	1.9E+06	1629.58	1324.45	1332.95
		Std	54.3	1.1E+04	3.4E+06	4.2E+05	118.2	1.9E+06	198.11	102.61	86.26
	f13	Ave	1305.92	8601.13	10828.4	11053.4	1310.4	16098.6	1323.63	1304.66	1303.74
		Std	2.58	5123.46	8928.9	2110.1	2.97	10537.2	78.32	0.71	3.26
	f14	Ave	1403.09	1482.09	7048.96	7147.73	1413.38	1508.94	1452.02	1410.85	1400.19
		Std	4.06	42.68	8160.08	1489.13	5.19	51.05	35.98	9.21	0.45
	f15	Ave	1500.77	1717.88	9296.19	18001.4	1502.30	2236.69	1509.65	1500.33	1500.33
		Std	0.52	282.34	8978.19	5498.7	0.60	571.19	16.43	0.36	0.20
	f16	Ave	1601.82	1860.07	1786.36	2149.84	1615.87	1726.26	1815.26	1602.50	1600.87
		Std	0.99	127.66	129.07	105.88	23.30	126.97	230.11	2.20	0.36
	f17	Ave	1714.55	1761.60	1746.54	1857.05	1730.45	1774.57	1830.07	1716.39	1701.37
		Std	9.44	47.51	39.78	108.33	4.08	34.23	175.80	5.96	3.84
	f18	Ave	1800.95	1.5E+04	1.6E+04	8720.53	1809.81	2.3E+04	1825.94	1809.87	1803.59
		Std	0.52	1.2+04	1.3E+04	5060.76	3.02	1.4E+04	13.53	9.47	7.60
	f19	Ave	1900.90	2602.84	9686.60	4.5E+04	1902.29	2916.15	1920.54	1900.52	1900.26
		Std	0.45	2185.03	6766.32	1.9E+04	0.46	1871.27	28.68	0.28	0.03
	f20	Ave	2015.52	2085.03	2056.55	2272.89	2030.97	2089.31	2494.47	2020.00	2000.23
		Std	9.67	62.25	60.02	81.73	6.21	49.28	242.65	0.01	0.43
Composition	f21	Ave	2203.72	2281.72	2303.81	2357.41	2214.96	2249.87	2324.76	2282.46	2255.42
		Std	20.35	54.02	43.75	28.21	30.48	60.44	67.76	42.64	52.16
	f22	Ave	2283.76	2314.80	2304.60	2300.01	2282.84	2301.52	3532.42	2297.27	2300.10
		Std	38.10	66.10	2.39	0.07	31.79	11.80	847.37	16.19	0.17
	f23	Ave	2611.63	2620.81	2632.89	2736.48	2619.05	2621.73	2728.84	2608.08	2602.30
		Std	3.93	9.24	13.42	39.14	3.31	8.69	243.09	1.71	1.42
	f24	Ave	2516.88	2692.19	2754.33	2742.17	2536.51	2733.20	2704.39	2728.92	2688.30
		Std	38.39	108.21	14.93	5.55	39.02	64.43	73.38	31.74	91.65
	f25	Ave	2897.92	2924.04	2947.94	2937.54	2861.00	2923.55	2932.01	2916.36	2923.77
		Std	0.49	25.02	19.25	15.31	89.81	23.86	20.87	22.94	21.30
	f26	Ave	2849.81	2952.13	3112.11	3440.66	2827.66	2900.96	3457.72	2909.17	2900.00
		Std	96.29	249.66	334.66	628.74	80.82	36.56	598.89	34.93	0.00
	f27	Ave	3089.37	3116.22	3115.14	3259.34	3090.40	3092.60	3137.46	3071.46	3089.03
		Std	0.46	24.99	19.19	41.66	0.95	2.78	210.37	0.78	1.05
	f28	Ave	3100.00	3315.89	3320.70	3459.45	3092.80	3210.58	3397.58	3266.65	3154.35
		Std	6.34E-05	121.84	126.35	33.84	61.60	113.17	131.30	22.26	110.92
	f29	Ave	3146.26	3203.06	3253.51	3449.28	3190.48	3214.18	3213.54	3142.84	3134.92
		Std	12.80	52.26	81.99	171.08	23.37	51.69	109.79	12.89	3.87
	f30	Ave	3414.92	3.5E+05	5.3E+05	9.4E+05	7038.71	4.2E+05	3.0E+05	3201.06	3418.38
		Std	26.81	5.1+05	6.3E+05	3.6E+05	3599.19	5.7E+05	4.4E+05	0.31	22.86

Friedman mean rank	2.60	5.74	7.14	7.76	4.03	6.67	6.34	2.71	2.00
Rank	2	5	8	9	4	7	6	3	1

6.6 Statistical Analysis of MPA

This section entails performing multiple statistical analysis, including Friedman, Bonferroni-Dunns and Holm's test, of MPA with respect to its other competitors. A simple non-parametric Friedman test showed that there is a statistically significant difference between the performance of the algorithms. In order to have a reliable test, this study categorized the benchmark functions into three groups. The first group is composed of functions shown in Table 2 which includes uni-modal, multimodal and composition functions. The second group is the class of CEC-BC-2017 functions shown in Table 3 and the third group is the combination of the first and second groups. After demonstrating significant differences among performance of different algorithms, there is a need to find out that which algorithms' performance are significantly different than MPA. Therefore, this study ran a post hoc statistical analysis of Bonferroni-Dunn. This test demonstrates that there is a significant difference in performance between two algorithms if the difference in average ranking of methods is greater than the critical difference (CD). MPA is considered as the control algorithm. **Error! Reference source not found.** shows the average ranking of each method in three groups of functions with two significance level of 0.1, 0.05. MPA is able to significantly outperform those algorithms whose average ranking is above the threshold line depicted in the figure. The threshold line of each group is identified by its color. As it is observable from the figure, MPA is ranked second with a very small difference in ranking following LSHADE-cnEpSin. It is possible to nominate both algorithms as a joint winner in this group, but strictly speaking, MPA ranked second in this group and could significantly outperform PSO, GA, GSA, CS, SSA and CMA-ES for both significance level of 0.1 and 0.05. For Group 2, EO was again ranked as second after LSHADE-cnEpSin and could significantly outperform PSO, GA, GSA, SSA, and CMA-ES. Finally, in group three, evaluating the performance of all functions, MPA ranked as the second-best performing algorithm among all its competitors with an average ranking of 2.59. The first rank in this class is achieved by LSHADE- cnEpSin with average ranking of 2.25. Again, in this class, MPA were able to show significantly better performance than all algorithms except SHADE and LSHADE-cnEpSin. Unfortunately, Bonferroni-Dunns test cannot specify the difference among the algorithms which their average ranks are below the threshold line. So, we tried another advanced post hoc analysis of Holm's test to first distinguished if there is a significant difference among the methods which placed below the threshold line and second confirms the Bonferroni-Dunns test results. Holm's method starts with sorting all algorithms based on their p-value and then compare it with the index of $\alpha/k - i$, where α is significance level, k is the degree of freedom and i is the algorithm number. The method sequentially rejects null hypothesis starting from the most significant p-value as long as $p_i < \alpha/k - i$. The algorithm stops as soon as the condition is not satisfied while considering all the remaining hypotheses as accepted. The results for group 1 in Table 4 confirm the Bonferroni-Dunns test results for both significance levels and also shows that there is no significance difference among the performance of MPA, SHADE, and LSHADE-cnEpSin. For group 2 functions, Table 5 confirms that MPA has significantly superior performance than PSO, GA, GSA, SSA, CMA-ES at both levels while its performance is similar to CS, SHADE, and LSHADE-cnEpSin. And finally, for all functions in group 3, shown in Table 6, Holm's test showed that MPA's performance is significantly better than PSO, GA, GSA, CS, SSA, CMA-ES while its performance is similar to SHADE and LSHADE- cnEpSin at both levels. To summarize this section, based on Figure 9 we can clearly observe that how MPA shows a reliable and confident behavior in all three groups of functions compared to other methods even high performance optimizer. The average ranking of MPA in all three groups is very close to each other demonstrating a reliable performing while other methods have unstable behavior in different groups of functions in terms of ranking.

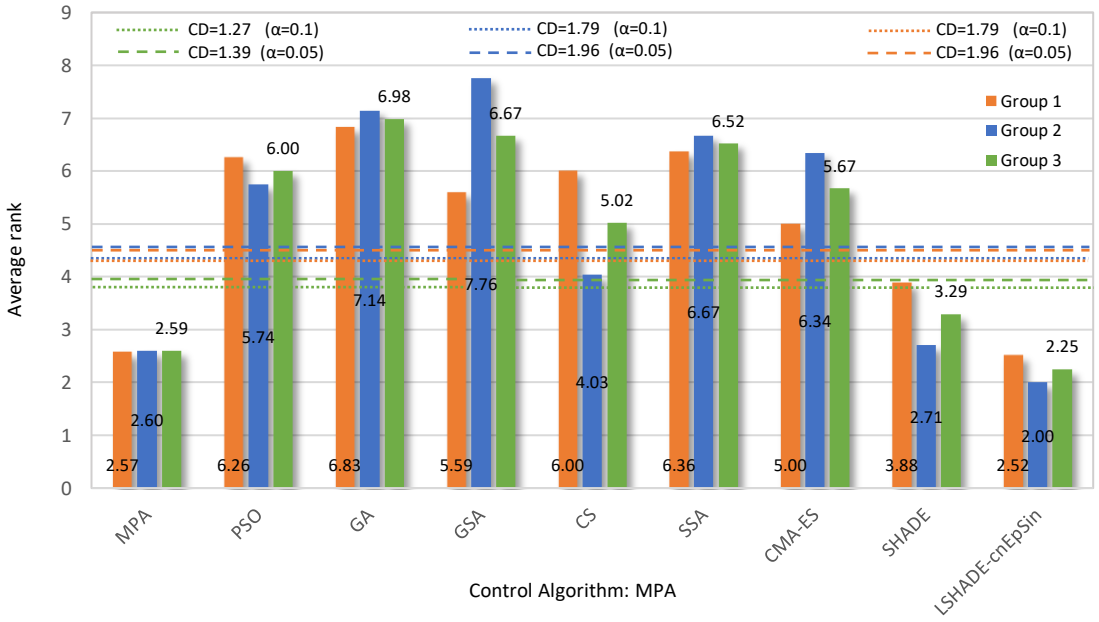


Figure 10. Bonferroni-Dunn's test for different methods and groups with $\alpha = 0.05$ and $\alpha = 0.1$

Table 4. Holm's statistical test for group 1 of functions (Control algorithm: MPA)

MPA vs.	rank	z-value	p-value	$\alpha/k-i$ (0.05)	$\alpha/k-i$ (0.1)
GA	6.82758	5.92137	3.86E-08	0.00625	0.0125
SSA	6.36206	5.27409	1.45E-06	0.00714	0.0142
PSO	6.25862	5.13025	3.08E-06	0.00833	0.0166
CS	6.00000	4.77066	1.83E-05	0.01	0.02
GSA	5.58620	4.19530	3.37E-05	0.0125	0.025
CMA-ES	5.00000	3.38021	7.24E-04	0.0166	0.0333
SHADE	3.87931	1.82196	6.85E-02	0.025	0.05
LSHADE-cnEpSin	2.51724	-0.07192	9.42E-02	0.05	0.1

Table 5. Holm's statistical test for group 2 of functions (Control algorithm: MPA)

MPA vs.	rank	z-value	p-value	$\alpha/k-i$ (0.05)	$\alpha/k-i$ (0.1)
GSA	7.75862	7.16797	1.09E-11	0.00625	0.0125
GA	7.13793	6.30494	3.68E-09	0.00714	0.0142
SSA	6.67241	5.65766	1.78E-07	0.00833	0.0166
CMA-ES	6.34482	5.20217	2.12E-06	0.01	0.02
PSO	5.74137	4.36311	1.62E-05	0.0125	0.025
CS	4.03448	1.98977	4.66E-02	0.0166	0.0333
LSHADE-cnEpSin	2.00000	-0.83906	4.01E-01	0.025	0.05
SHADE	2.70689	0.14383	8.86E-01	0.05	0.1

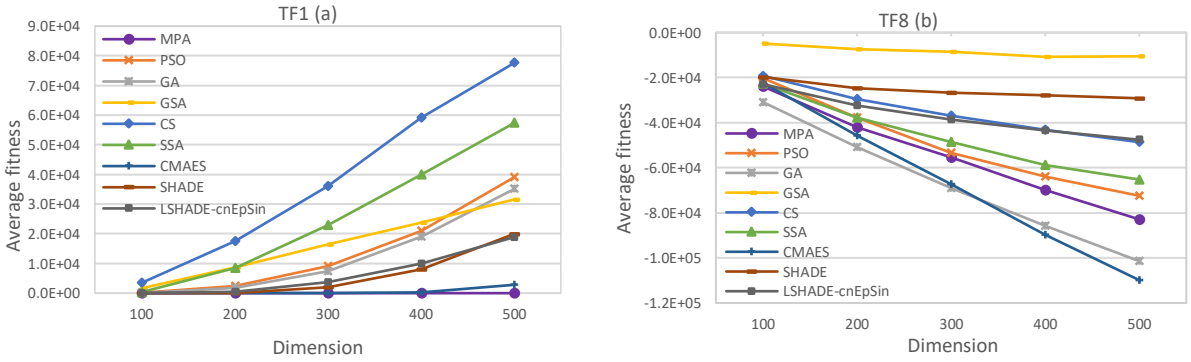
Table 6. Holm's statistical test for group 3 of functions (Control algorithm: MPA)

MPA vs.	rank	z-value	p-value	$\alpha/k-i$ (0.05)	$\alpha/k-i$ (0.1)
GA	6.98275	6.10117	1.31E-08	0.00625	0.0125
GSA	6.67241	5.66965	1.66E-07	0.00714	0.0142
SSA	6.51724	5.45389	5.54E-07	0.00833	0.0166
PSO	6.00000	4.73470	2.17E-05	0.01	0.02
CMA-ES	5.67241	4.27921	2.35E-05	0.0125	0.025
CS	5.01724	3.36823	7.56E-04	0.0166	0.0333
SHADE	3.29310	0.97091	3.32E-01	0.025	0.05
LSHADE-cnEpSin	2.25000	-0.47946	6.32E-01	0.05	0.1

6.7 Scalability analysis of MPA

This section evaluates the performance of the proposed MPA method in high dimensional problems using scalability analysis with respect to other methods. In order to have a concise and efficient comparison, the test is conducted with two functions from group of 1 and two functions from group 2. The first functions from each category of unimodal (TF1) and multimodal (TF8) of group 1 is selected for the test. Since the fix-dimensional and composition test functions from group 1 are specifically designed for a fix dimension, we cannot use them for scalability analysis. For group 1 of functions, the algorithm starts from 100 to 500 dimensions with step size of 100. From group 2 we selected the first functions of hybrid (f11) and composition function (f21). The group 2 functions are only designed for the dimension of 10, 20, 30, 50 and 100 so we are only able to conduct the analysis in these dimensions. The maximum number of function evaluations is considered fixed for each group similar to their original test conditions; 25,000 for group 1 and 50,000 for group 2 with 30 independent runs.

Figure 11 shows the performance of methods as the dimension increase. X-axis shows the dimension and Y-axis shows the average fitness calculated from different runs. In Figure 11 (a), MPA showed the best performance in TF1 compared to others with achieving the lowest average fitness as the dimension increase. In TF8 Figure 11 (b), all methods followed an almost linear pattern with different slopes. As it is observable, MPA ranked third in most dimensions following CMAES and GA. In Figure 11 (c), f11, most methods have almost similar behavior till dimension 50 except GA and GSA, but in the last dimension of 100, the performance of each method became more obvious. In this dimension, EO ranked as the third-best performing method following the best methods as LSHADE-cnEpSin and SHADE. In the last function, f21, MPA again get the third place after CMAES and LSHADE-cnEpSin which ranked first and second respectively. It is interesting to know that these ranks are more or less similar to what MPA could obtain compared to other methods in its numerical evaluation. This behavior clearly demonstrates that MPA is a reliable algorithm and its performance remains high and very competitive to CEC competition winners dealing the high dimensional problems with a limited budget.



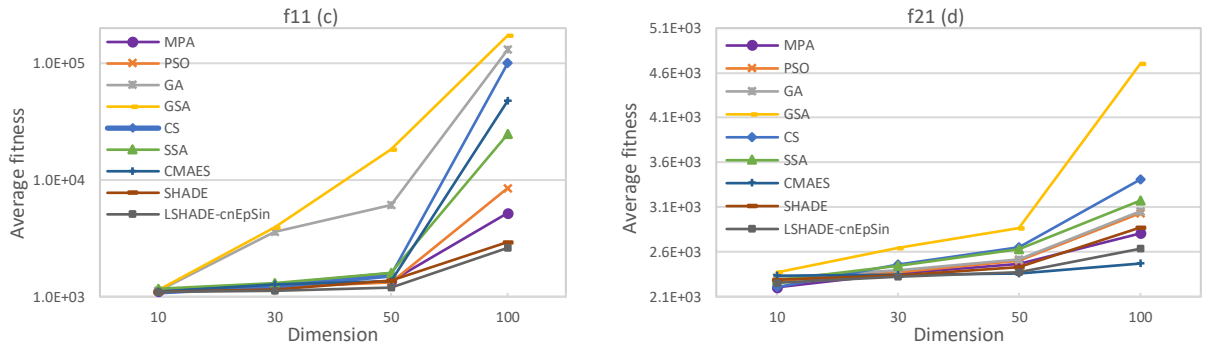


Figure 11. Scalability analysis of different methods

7. Engineering design problems

This section assesses the performance of MPA in real-world problems using constrained engineering benchmarks. The engineering problem includes pressure vessel design, welded beam design, and tension/compression spring design along with a practical engineering example. A simple method of death penalty is employed here to convert the constrained problems into the unconstrained ones. Similar to the mathematical test functions, during the solution of the engineering design problems, MPA uses the same number of iteration (500), maximum function evaluation (25,000) and run (30) unless it is reported differently. The mathematical formulation of the problem can be found in (Mirjalili et al., 2014).

7.1 Pressure vessel design

This problem is one of the widely used mixed-integer design problems. The objective is to minimize the total cost of welding, material, and forming of the pressure vessel shown in Figure 12. The design variables include the thickness of the shell (Ts), the thickness of the head (T_h), the inner radius (R), and the length of the cylindrical section of the vessel (L). Based on the ASME boiler code requirement, the available thickness for the shell (T_s) and head (T_h) are integer multiples of 0.0625 inch. It should be noted that some literature reported the results in continuous optimization.

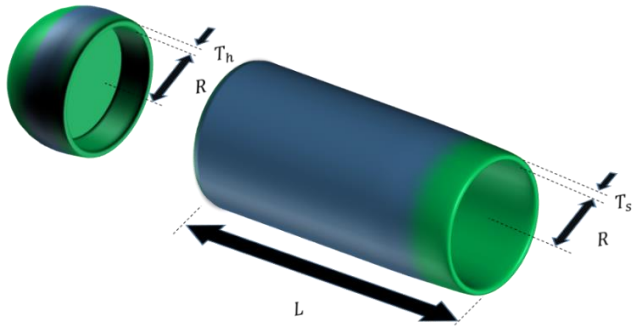


Figure 12. Pressure vessel

Existing researchers have investigated this problem with various metaheuristic algorithms such as GA (Bernardino, Barbosa, Lemonge, & Fonseca, 2008), PSO (Q. He & Wang, 2007), ES (Efrén Mezura-Montes & Coello, 2008), HS (Lee & Geem, 2005), and DE (Huang, Wang, & He, 2007). Table 7 shows the optimum results achieved by MPA and its comparison to those of other methods

while Table 8 includes the associated statistical results. Since a couple of studies have considered the problem as continuous, the results achieved by MPA are shown in both mixed-integer and continuous. Inspecting the results in Table 7, it is evident that MPA is able to outperform other methods by achieving the lowest total cost among others. Interestingly, while MPA terminated the optimization in 25,000 function evaluation, the statistical results reported in Table 8 is better compared to other methods. Overall, the results of this study show the effectiveness and efficiency of MPA to solve this problem.

Table 7. Optimum results of pressure vessel problem by different algorithms

Algorithm	$T_s(x_1)$	$T_h(x_2)$	$R(x_3)$	$L(x_4)$	f_{cost}
GSA	1.1250000	0.625000	55.9886598	84 .4542025	8538 .8359
HS (Lee & Geem, 2005)	1.125	0.625	58.2789	43.7549	7198.433
HGA(2) (Bernardino et al., 2008)	1.1250	0.5625	58.1267	44.5941	6832.583
GeneAS (Deb, 1997)	0.9375000	0.500000	48.329000	112.679000	6410.3811
T-Cell (Aragón, Esquivel, & Coello, 2010)	0.8125	0.4375	42.098429	190.787695	6390.554
SBM (Akhtar, Tai, & Ray, 2002)	0.8125	0.4375	41.9768	182.2845	6171.000
HGA(1) (Bernardino et al., 2008)	0.8125	0.4375	42.0492	177.2522	6065.821
CPSO (Q. He & Wang, 2007)	0.8125	0.4375	42.091266	176.746500	6061.0777
BFOA (Montes & Ocana, 2008)	0.8125	0.4375	42.096394	176.683231	6060.460
HAIS-GA (C. A. C. Coello & Cortés, 2004)	0.8125	0.4375	42.0931	176.7031	6060.367
DTS-GA (C. Coello & Montes, n.d.)	0.8125	0.4375	42.097398	176.654047	6059.9463
ES (Efrén Mezura-Montes & Coello, 2008)	0.8125	0.4375	42.098087	176.640518	6059.745
CDE (Huang et al., 2007)	0.8125	0.4375	42.098411	176.637690	6059.7340
MPA (Mixed-integer)	0.8125	0.4375	42.098445	176.636607	6059.7144
MPA (Continuous)	0.77816876	0.38464966	40.31962084	199.9999935	5885.3353

Table 8. Statistical results of pressure vessel problem by different algorithms

Algorithm	Best	Mean	Worst	Std	Eval, No
HGA(2) (Bernardino et al., 2008)	6832.584	7187.314	8012.615	276	80,000
T-Cell (Aragón et al., 2010)	6390.554	6737.065	7694.066	357	80,000
SBM (Akhtar et al., 2002)	6171.00	6335.05	6453.65	N.A	12,630
HGA(1) (Bernardino et al., 2008)	6065.821	6632.376	8248.003	515	80,000
CPSO (Q. He & Wang, 2007)	6061.0777	6147.1332	6363.8041	86.4545	200,000
BFOA (Montes & Ocana, 2008)	6060.460	6074.625	N.A	156	48,000
HAIS-GA (C. A. C. Coello & Cortés, 2004)	6061.1229	6743.0848	7368.0602	457.99	150,000
DTS-GA (C. Coello & Montes, n.d.)	6059.9463	6177.2532	6469.322	130.92	80,000
ES (Efrén Mezura-Montes & Coello, 2008)	6059.746	6850.00	7332.87	426	25,000

CDE (Huang et al., 2007)	6059.7340	6085.2303	6371.0455	43.013	240,000
MPA	6059.7144	6102.8271	6410.0929	106.61	25,000

7.2 Welded beam design

The second example to evaluate the performance of MPA in the engineering domain is a well-known welded beam design shown in Figure 13. The objective is to find the best design variables to minimize the total fabrication cost of a welded beam subject to shear stress(τ), bending stress(σ), buckling load(P_c), deflection (δ) and other constraints. Considering $x_1 = h,\ x_2 = l,\ x_3 = t,\ x_4 = b$, as design variables, the mathematical formulation is expressed as follows:

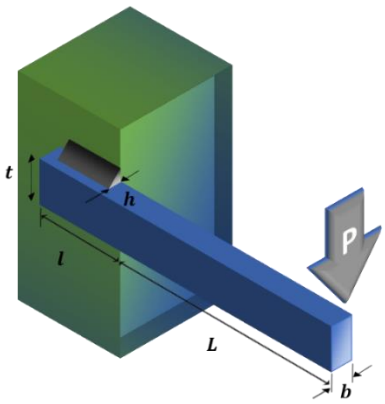


Figure 13. Welded beam

Existing studies have investigated this problem: (i) Atiqullah and Rao (Atiqullah & Rao, 2000) utilized SA; (iii) He and Prempain (S. He, Prempain, & Wu, 2004) deployed PSO method; (iv) Zhang et al. (M. Zhang, Luo, & Wang, 2008) used DE. Table 9 compares the optimum results by MPA with those obtained by different algorithms documented in the literature. It is seen that the best solution achieved by MPA is better than those quoted for other methods. The corresponding statistical results reported in Table 10 also validate the extensive exploration ability of MPA in which the best, mean, worst and standard deviation (Std) of MPA are reported far better than other literature. The significantly low standard deviation achieved by MPA proves that the proposed algorithm is more robust than other optimization methods.

Table 9. Optimum results of welded beam problem by different algorithms

Algorithm	$h(x_1)$	$l(x_2)$	$t(x_3)$	$b(x_4)$	f_{cost}
SBM (Akhtar et al., 2002)	0.2407	6.4851	8.2399	0.2497	2.4426
SA (Atiqullah & Rao, 2000)	0.2471	6.1451	8.2721	0.2495	2.4148
BFOA (Montes & Ocana, 2008)	0.2057	3.4711	9.0367	0.2057	2.3868
SCA (Ray & Liew, 2003)	0.2444	6.2380	8.2886	0.2446	2.3854
EA (J. Zhang, Liang, Huang, Wu, & Yang, 2009)	0.2443	6.2201	8.2940	0.2444	2.3816
T-Cell (Aragón et al., 2010)	0.2444	6.1286	8.2915	0.2444	2.3811
FSA (Hedar & Fukushima, 2006)	0.2444	6.1258	8.2939	0.2444	2.3811
IPSO (S. He et al., 2004)	0.2444	6.2175	8.2915	0.2444	2.3810

DSS-DE (M. Zhang et al., 2008)	0.2444	6.1275	8.2915	0.2444	2.3810
HS (Lee & Geem, 2005)	0.2442	6.2231	8.2915	0.2443	2.3807
HSA-GA (Hwang & He, 2006)	0.2231	1.5815	12.8468	0.2245	2.2500
GSA	0.182129	3.856979	10.0000	0.202376	1.879952
RO (A. Kaveh & Khayatazad, 2012)	0.203687	3.528467	9.004233	0.207241	1.735344
CDE (Huang et al., 2007)	0.203137	3.542998	9.033498	0.206179	1.733462
CPSO (Q. He & Wang, 2007)	0.202369	3.544214	9.048210	0.205723	1.728024
MPA	0.205728	3.470509	9.036624	0.205730	1.724853

Table 10. Statistical results of welded beam problem by different algorithms

Algorithm	Best	Mean	Worst	Std	Eval, No
SBM (Akhtar et al., 2002)	2.4426	2.5215	2.6315	N.A	19,259
BFOA (Montes & Ocana, 2008)	2.3868	2.4040	N.A	0.016	48,000
SCA (Ray & Liew, 2003)	2.3854	3.2551	6.3996	0.9590	33,095
EA (J. Zhang et al., 2009)	2.3816	N.A	2.38297	0.00034	28,897
T-Cell (Aragón et al., 2010)	2.3811	2.4398	2.7104	0.09314	320,000
FSA (Hedar & Fukushima, 2006)	2.3811	2.4041	2.4889	N.A	56,243
IPSO (S. He et al., 2004)	2.3810	2.3819	N.A	0.00523	30,000
HSA-GA (Hwang & He, 2006)	2.2500	2.26	2.28	0.0078	26,466
RO (A. Kaveh & Khayatazad, 2012)	1.735344	1.9083	N.A	0.173744	8,000
CDE (Huang et al., 2007)	1.733461	1.768158	1.824105	0.022194	240,000
CPSO (Q. He & Wang, 2007)	1.728024	1.748831	1.782143	0.012926	200,000
MPA	1.724853	1.724861	1.724873	6.41E-06	25,000

7.3 Tension/compression spring design

The objective of this problem is to find the minimum weight of a tension/compression spring illustrated in Figure 14. The optimization constraints are defined as minimum deflection, shear stress and surge frequency along with design variables including wire diameter(d), mean coil diameter (D) and number of active coils(N). The mathematical formulation of the problem is as follows:

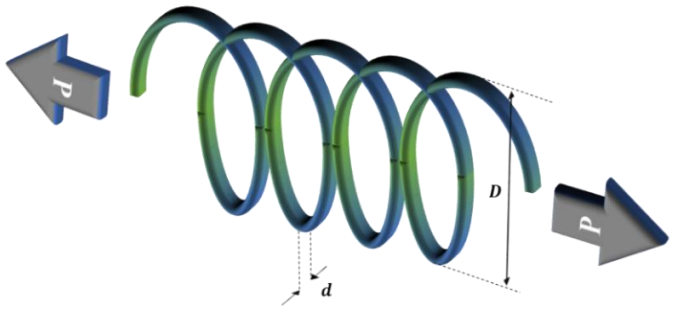


Figure 14. Tension/compression spring design

This problem has been solved by GA (Coello Coello, 2000), Evolution Strategy (ES) (E. Mezura-Montes & Coello, 2005), PSO (Q. He & Wang, 2007) and Differential Evolution (DE) (Huang et al., 2007). Table 11 presents the optimum design and corresponding cost function (weight) obtained by MPA and other algorithms. It can be seen that again MPA was able to get the best results among other methods.

Table 11. Optimum results of spring design problem by different algorithms

Algorithm	$d(x_1)$	$D(x_2)$	$N(x_3)$	f_{cost}
GA(1) (Coello Coello, 2000)	0.051480	0.351661	11.632201	0.012704
CA (C. A. C. Coello & Becerra, 2004)	0.050000	0.317395	14.031795	0.012721
GSA	0.050276	0.323680	13.525410	0.012702
GA(2) (Coello Coello & Mezura Montes, 2002)	0.051989	0.363965	10.890522	0.012681
ES (E. Mezura-Montes & Coello, 2005)	0.051643	0.355360	11.397926	0.012698
CPSO (Q. He & Wang, 2007)	0.051728	0.357644	11.244543	0.012674
BFOA (Montes & Ocana, 2008)	0.051825	0.359935	11.107103	0.012671
CDE (Huang et al., 2007)	0.051609	0.354714	11.410831	0.012670
SCA (Ray & Liew, 2003)	0.052160	0.368159	10.648442	0.012669
HGA (Bernardino, Barbosa, & Lemonge, 2007)	0.051302	0.347475	11.852177	0.012668
PFA (Yapici & Cetinkaya, 2019)	0.051726	0.357629	11.235724	0.012665
T-Cell (Aragón et al., 2010)	0.051622	0.355105	11.384534	0.012665
MPA	0.051724477	0.35757003	11.2391955	0.012665

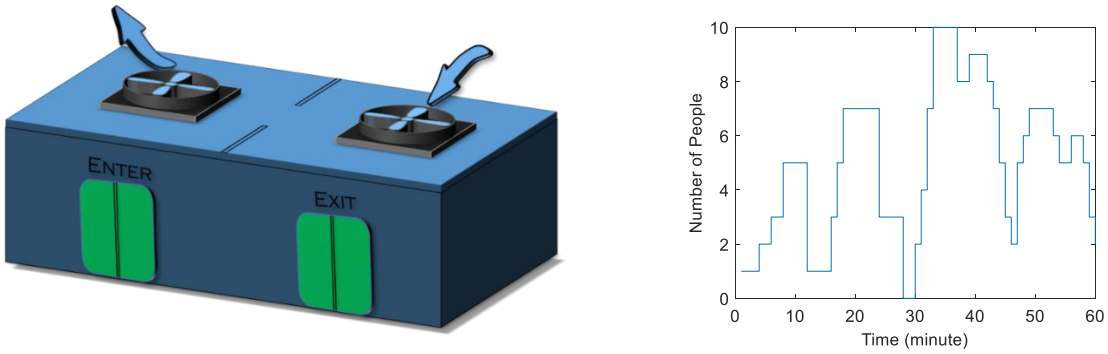
The statistical information regarding the design process of each method is illustrated in Table 12. Based on the table, MPA again outperformed other methods regarding best, mean, worst and standard deviation. MPA required only 25,000 function evaluations to accomplish the optimization which highlights the high efficiency of the proposed algorithm.

Table 12. Statistical results of spring design problem by different algorithms

Algorithm	Best	Mean	Worst	Std	Eval, No
GA(1) (Coello Coello, 2000)	0.012704	0.012769	0.012822	3.93E-05	N/A
CA (C. A. C. Coello & Becerra, 2004)	0.012721	0.013568	0.0151156	8.4E-04	50,000
GA(2) (Coello Coello & Mezura Montes, 2002)	0.012681	0.012742	0.012973	9.5E-05	80,000
CPSO (Q. He & Wang, 2007)	0.012674	0.012730	0.012924	5.19E-05	200,000
BFOA (Montes & Ocana, 2008)	0.012671	0.012759	N/A	1.36E-04	48,000
CDE (Huang et al., 2007)	0.012670	0.012703	0.012790	2.07E-05	240,000
SCA (Ray & Liew, 2003)	0.012669	0.012922	0.016717	5.92E-04	25,167
HGA (Bernardino et al., 2007)	0.012668	0.013481	0.016155	N/A	36,000
T-Cell (Aragón et al., 2010)	0.012665	0.013309	0.012732	9.4E-05	36,000
MPA	0.012665	0.012665	0.012665	5.55E-08	25,000

7.4 Operating fan schedule for demand-controlled ventilation

This section considers a practical example of a two-zone retail store equipped with supply and exhaust fan for ventilation purposes to further challenge the MPA performance in real-world optimization. The objective is to minimize the fan energy consumption using demand control ventilation under the constraints of airflow and zone CO_2 concentration. Based on the affinity law in ventilation studies, the power draw of a fan is proportional to the cube of its flow rate. The fans are actuated based on a signal received from Variable Frequency Drives (VFDs) to regulate the airflow while keeping the CO_2 concentration under the defined threshold. **Error! Reference source not found.** shows the schematic of a small store and its associated occupancy schedule. Two zones are considered here as "enter" and "exit" zones having a dimension of 5m length, 5m width, and 3m height.



(a) (b)

Figure 15. Retail store schematic (a) and its occupancy schedule (b)

The objective function of the problem is defined as the energy consumption of operating fans for the period of one hour. Q (m^3/min) and W (Watt) are the airflow and power draw. There is a reference condition for operating fans of which the power draw is calculated for other magnitudes of airflows. This reference condition is considered as the airflow and its associated power draw at full motor capacity; $Q_{ref} = 6$ (m^3/min) and $W_{ref} = 46$ ($Watt$). CO_2 concentration in each zone is calculated using the mass-balance equation presented by a simple ODE.

$$\frac{dC}{dt} = \frac{Q}{V}(C_{out} - C) + \frac{E}{V} \tag{17}$$

C stands for concentration (mg/m^3), t is a time in minute, V is the volume of each zone, C_{out} is outdoor CO_2 concentration considered as 775(mg/m^3), E is the emission rate of CO_2 due to occupancy which is $E = 569.2$ ($mg/min/person$). Initial conditions for CO_2 concentration are assumed as $C_0 = 980$ and $C_0 = 1{,}000$ (mg/m^3) for exit and enter zones, respectively.

$$\begin{aligned}
&Consider && \vec{Q} = [Q_1 Q_2 \dots Q_{60}] \\
&Minimize && f_{cost}(\vec{Q}) = \frac{1}{60} \sum_{supply}^{Exhaust} \sum_{i=1}^{60} W_{ref} \left(\frac{Q}{Q_{ref}}\right)^3 \qquad (18) \\
&Subject\ to && g_1(\vec{x}) = 0 < C_{enter}, C_{exit} < 1960 \quad (mg/m^3) \\
&Variable\ range && 0 \le Q_1, \dots, Q_{60} \le 6 \ \ m^3/min
\end{aligned}$$

Error! Reference source not found. presents the optimal operating fan schedule derived by MPA, PSO, and GA. It is observable that in the optimal schedule achieved by GA and PSO, there are a handful of abrupt changes in airflow during the operating period. These sudden changes in fan motor speed will lead to structural problems, e.g., fatigue and failure, and will reduce the performance of the fans in the long run. As **Error! Reference source not found.**(a) shows, MPA proposes a smooth yet efficient schedule by increasing the motor speed uniformly compared to PSO and GA methods illustrated in **Error! Reference source not found.**(b) and **Error! Reference source not found.**(c).

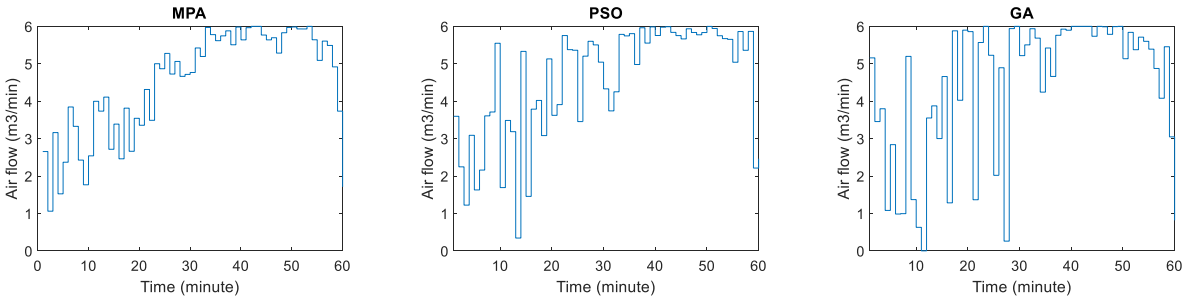


Figure 16. Optimal operating fan schedule for MPA (a) PSO (b) and GA (c)

Table 13 shows the statistical results for fan energy consumption in Watt-hour. MPA could outperform other methods by achieving a better energy efficient schedule while presenting more reliable solutions by achieving a lower standard deviation.

Table 13. Statistical results for fan energy consumption in Watt-hour obtained by different methods

Algorithm	Best	Mean	Worst	Std	Eval, No
MPA	49.29	50.11	52.00	0.66	15,000
PSO	52.10	55.75	58.97	2.21	15,000
GA	53.45	55.46	58.34	1.83	15,000

7.5 Building energy performance

This example deploys MPA in one of the common building science engineering problems to minimize the total annual energy consumption of an office building. The energy consumption of a blue shaded thermal zone is considered as a representative of an elongated office building located in Chicago, IL. There are east and west windows associated with an external shading device which is activated during summer when total solar irradiation exceeds 200 (W/m^2). Dimension and geometry of office, windows and shading are shown in the figure. The zone has daylighting control with illuminance set-point of 500 lux. Ideal load air systems as selected for the Heating, Ventilation, and Air-Conditioning (HVAC) system meaning that the loads (heating and cooling) are assumed to be met in each time step of the simulation. Heating and cooling set-points are adjusted as 20°C and 25°C, respectively. Exterior walls have a U-value of 0.25 ($W/m^2\,k$) with 10 cm of foam insulation defined for the baseline case. Detailed information about the construction and materials can be found in (Wetter, 2004).

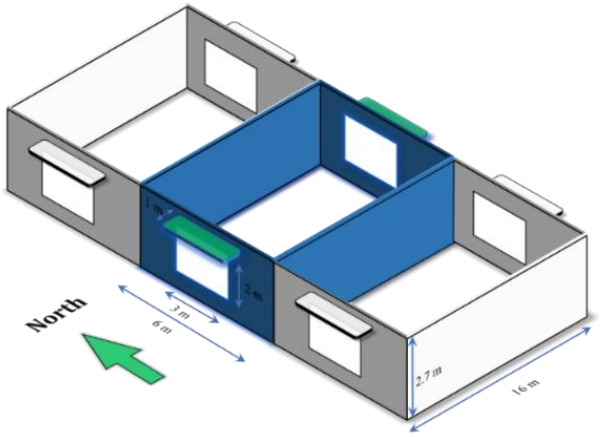


Figure 17. Schematic of the office building

The objective function is considered as annual source energy consumption of the typical office room presented as:

$$f_{cost}(\vec{x}) = \frac{Q_h(x)}{\eta_h} + \frac{Q_c(x)}{\eta_c} + \alpha E_l(x) \tag{19}$$

Q_h, Q_c are the annual heating, cooling load and E_l is the electricity consumption for the zone under study. $\eta_h = 0.44, \eta_c = 0.77$ are typical plant efficiencies. $\alpha = 3$ is the factor that converts the site electricity to fuel energy consumption. The design variables and lower and upper bounds are shown in Table 14. The optimization was accomplished with 10 search agents and 100 iterations due to high computational run time the average results are achieved by 10 runs.

Table 14. Building energy performance optimization variables and their related thresholds

Variables	Units (SI)	Notations	Lower bound	Upper bound
Orientation	degree	x_1	0	360
Shading solar transmittance	dimensionless	x_2	0	0.9
West window length	meter	x_3	0	5.9
East window length	meter	x_4	0	5.9
Insulation thickness	meter	x_5	0.01	0.5
Insulation conductivity	watts/(meter-kelvin)	x_6	0.01	1
Glazing solar transmittance	dimensionless	x_7	0	0.28

Table 15 presents the annual source energy consumption for the office building achieved by MPA, PSO, and GA. Comparing the best, mean and standard deviation results shows that MPA and GA reach almost the same results in this example, which is better than that of PSO.

Table 15. Source energy consumption (Joule) and its statistical results obtained by different methods

Algorithm	Best	Mean	Worst	Std	Eval, No
MPA	4.48E10	4.49E10	4.51E10	8.92E07	1000
PSO	4.64E10	4.95E10	5.38E10	2.51E09	1000
GA	4.48E10	4.48E10	4.51E10	9.92E07	1000

The solution associated with the best design for each optimization method is brought in Table 16. Since MPA and GA reached almost the same energy consumption, the optimal design variables should be close to each other as well. Referring to Table 16, the MPA and GA design parameters are the same with 180° difference in orientation in which the west window in MPA's design can be considered as the east window in GA's design.

Table 16. Optimal solutions for building energy performance example obtained by different methods

Variables	MPA	PSO	GA
Orientation	85.96	258.3	265.9
Shading solar transmittance	0	0.3	0
West window length	5.9	4.26	4.49
East window length	4.51	5.08	5.9
Insulation thickness	0.45	0.48	0.5
Insulation conductivity	0.01	0.11	0.01
Glazing solar transmittance	0.28	0.18	0.28

It is important to note that EnergyPlus (EnergyPlus, 2015) and jEplus (Y. Zhang, 2009) are employed as a simulation engine and interface. The simplified client-server architecture for this example is shown in Figure 18. Summarily, jEplus acts as a middleware platform which uses EnergyPlus engine for simulation. Optimizer receives output file containing energy consumption as objective function and design variables from EnergyPlus, then it optimizes the solution, and write different inputs files for jEplus. The loop is continued until the optimizer terminates the process based on its defined criterion. The results of this engineering optimization co-simulation have implications for the future design of energy-efficient buildings.

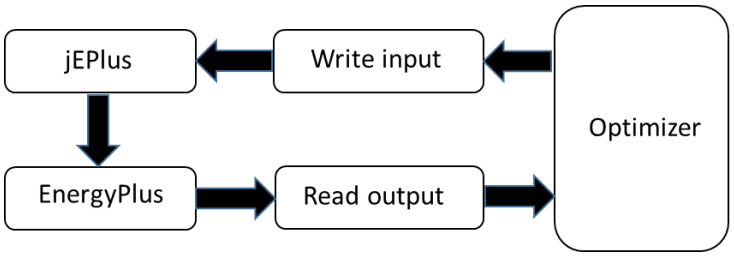


Figure 18. Client-server architecture of co-simulation for building energy performance

8. Conclusion

This paper proposed a novel and real nature-inspired optimization method, called Marine Predator Algorithm (MPA). The source of inspiration was based on different foraging strategy among ocean predators and optimal encounter rates policy in biological interaction. Lévy and Brownian motion are two types of strategies chosen by marine predators for optimal foraging. Based on the movement type and velocity of prey there is an optimal movement policy for a predator (Lévy or Brownian) in a way that maximizes its encounter rate with prey. From the movement point of view and in MPA, the lifetime of a predator with respect to prey was divided into three phases of (i) when the predator was moving faster than prey, (ii) when prey was moving faster than predator, and (iii) when both were moving at almost the same pace. In each phase, the optimal movement policy was attributed to predator to specify the step size taken to reach prey. The design of MPA mimics the rules and behavior of marine predator's foraging strategy to possibly present a real nature-inspired metaheuristic which its mathematical model is as close as to its natural model.

MPA's performance was evaluated against exploration, exploitation and local minima avoidance with 58 mathematical benchmark functions and the results were compared to those of three classes of optimization methods. i) GA and PSO as the most well-known and traditional optimization methods ii) GSA, CS and SSA as an almost recently developed metaheuristics and iii) CMA-ES,

SHADE, and LSHADE-cnEpSin as the high performance optimizers. The comprehensive post hoc analysis revealed that MPA is a significantly better optimizer than PSO, GA, GSA, CS, SSA and CMEAS, and its performance is statistically similar to SHADE and LSHADE-cnEpSin. To further challenge the performance of the proposed algorithm, MPA was also applied to three engineering benchmarks and two real-world design tasks. In engineering benchmarks, MPA obtained better or equal solutions with low computational cost compared to other methods. Thus, MPA proves its ability to present the most effective design and efficient statistical results compared to other methods.

The simplicity, ease of application along with its effective and efficient results can highlight the MPA as an alternative optimization algorithm to classical methods. Solving other optimization problems in different disciplines is recommended for more evaluation of MPA. Since MPA is a velocity-based algorithm developing a binary and multi-objective version of MPA would be a valuable contribution.

References

Akhtar, S., Tai, K., & Ray, T. (2002). A socio-behavioural simulation model for engineering design optimization. *Engineering Optimization*, *34*(4), 341–354. https://doi.org/10.1080/03052150212723

Aragón, V. S., Esquivel, S. C., & Coello, C. A. C. (2010). A modified version of a T-Cell Algorithm for constrained optimization problems. *International Journal for Numerical Methods in Engineering*, *84*(3), 351–378. https://doi.org/10.1002/nme.2904

Atiqullah, M. M., & Rao, S. S. (2000). Simulated Annealing and Parallel Processing: An Implementation for Constrained Global Design Optimization. *Engineering Optimization*, *32*(5), 659–685. https://doi.org/10.1080/03052150008941317

Awad, N H, Ali, M. Z., Suganthan, P. N., Liang, J. J., & Qu, B. Y. (2017). Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. *2017 IEEE Congress on Evolutionary Computation (CEC)*.

Awad, Noor H., Ali, M. Z., & Suganthan, P. N. (2017). Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems. *2017 IEEE Congress on Evolutionary Computation (CEC)*, 372–379. https://doi.org/10.1109/CEC.2017.7969336

Bartumeus, F., Catalan, J., Fulco, U. L., Lyra, M. L., & Viswanathan, G. M. (2002). Optimizing the encounter rate in biological interactions: Lévy versus Brownian strategies. *Physical Review Letters*, *88*(9), 097901. https://doi.org/10.1103/PhysRevLett.88.097901

Bartumeus, Frederic. (2007). Lévy processes in animal movement: an evolutionary hypothesis. *Fractals*, *15*(02), 151–162. https://doi.org/10.1142/S0218348X07003460

Bartumeus, Frederic, da Luz, M. G. E., Viswanathan, G. M., & Catalan, J. (2005). Animal Search Strategies: A Quantitative Random-Walk Analysis. *Ecology*, *86*(11), 3078–3087. https://doi.org/10.1890/04-1806

Bernardino, H. S., Barbosa, H. J. C., & Lemonge, A. C. C. (2007). A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. *2007 IEEE Congress on Evolutionary Computation*, 646–653. https://doi.org/10.1109/CEC.2007.4424532

Bernardino, H. S., Barbosa, H. J. C., Lemonge, A. C. C., & Fonseca, L. G. (2008). A new hybrid AIS-GA for constrained optimization problems in mechanical engineering. *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 1455–1462. https://doi.org/10.1109/CEC.2008.4630985

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge, U.K.: Cambridge University Press.

Clark, E. (1959). Instrumental Conditioning of Lemon Sharks. *Science*, *130*(3369), 217–218. https://doi.org/10.1126/science.130.3369.217-a

Coello, C. A. C., & Becerra, R. L. (2004). Efficient evolutionary optimization through the use of a cultural algorithm. *Engineering Optimization*, *36*(2), 219–236. https://doi.org/10.1080/03052150410001647966

Coello, C. A. C., & Cortés, N. C. (2004). Hybridizing a genetic algorithm with an artificial immune system for global optimization. *Engineering Optimization*, *36*(5), 607–634. https://doi.org/10.1080/03052150410001704845

Coello, C., & Montes, E. (n.d.). Use of dominance-based tournament selection to handle constraints in genetic algorithms. *Intelligent Engineering Systems through Artificial Neural Networks (ANNIE2001)*, *11*, 177–182. St. Louis, Missouri: ASME press.

Coello Coello, C. A. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, *41*(2), 113–127. https://doi.org/10.1016/S0166-3615(99)00046-9

Coello Coello, C. A., & Mezura Montes, E. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, *16*(3), 193–203. https://doi.org/10.1016/S1474-0346(02)00011-3

Deb, K. (1997). GeneAS: A Robust Optimal Design Technique for Mechanical Component Design. In *Evolutionary Algorithms in Engineering Applications* (pp. 497–514). https://doi.org/10.1007/978-3-662-03423-1_27

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39. https://doi.org/10.1109/MCI.2006.329691

Dugatkin, L. A., & Wilson, D. S. (1992). The prerequisites for strategic behaviour in bluegill sunfish, Lepomis macrochirus. *Animal Behaviour*, *44*(Part 2), 223–230. https://doi.org/10.1016/0003-3472(92)90028-8

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. , *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS '95*, 39–43. https://doi.org/10.1109/MHS.1995.494215

Einstein, A. (1956). *Investigations on the Theory of the Brownian Movement*. New York: Dover.

EnergyPlus (Version 8.0.1). (2015). U.S. Department of Energy (DOE).

Faramarzi, A., & Afshar, M. H. (2012). Application of cellular automata to size and topology optimization of truss structures. *Scientia Iranica*, *19*(3), 373–380. https://doi.org/10.1016/j.scient.2012.04.009

Faramarzi, A., & Afshar, M. H. (2014). A novel hybrid cellular automata–linear programming approach for the optimal sizing of planar truss structures. *Civil Engineering and Environmental Systems*, *31*(3), 209–228. https://doi.org/10.1080/10286608.2013.820280

Filmalter, J. D., Dagorn, L., Cowley, P. D., & Taquet, M. (2011). First Descriptions of the Behavior of Silky Sharks, Carcharhinus Falciformis, Around Drifting Fish Aggregating Devices in the Indian Ocean. *Bulletin of Marine Science*, *87*(3), 325–337. https://doi.org/10.5343/bms.2010.1057

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

Hansen, N., Müller, S. D., & Koumoutsakos, P. (2003). Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evol. Comput.*, *11*(1), 1–18. https://doi.org/10.1162/106365603321828970

He, Q., & Wang, L. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, *20*(1), 89–99. https://doi.org/10.1016/j.engappai.2006.03.003

He, S., Prempain, E., & Wu, Q. H. (2004). An improved particle swarm optimizer for mechanical design optimization problems. *Engineering Optimization*, *36*(5), 585–605. https://doi.org/10.1080/03052150410001704854

Hedar, A.-R., & Fukushima, M. (2006). Derivative-Free Filter Simulated Annealing Method for Constrained Continuous Global Optimization. *Journal of Global Optimization*, *35*(4), 521–549. https://doi.org/10.1007/s10898-005-3693-z

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.

Huang, F., Wang, L., & He, Q. (2007). An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and Computation*, *186*(1), 340–356. https://doi.org/10.1016/j.amc.2006.07.105

Humphries, N. E., Queiroz, N., Dyer, J. R. M., Pade, N. G., Musyl, M. K., Schaefer, K. M., … Sims, D. W. (2010). Environmental context explains Lévy and Brownian movement patterns of marine predators. *Nature*, *465*(7301), 1066. https://doi.org/10.1038/nature09116

Hwang, S. F., & He, R. S. (2006). A hybrid real-parameter genetic algorithm for function optimization. *Advanced Engineering Informatics*, *20*(1), 7–21. https://doi.org/10.1016/j.aei.2005.09.001

Jain, M., Singh, V., & Rani, A. (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm and Evolutionary Computation*, *44*, 148–175. https://doi.org/10.1016/j.swevo.2018.02.013

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

Kaveh, Ali, & Bakhshipour, T. (2019). *Metaheuristics: Outlines, MATLAB Codes and Examples* (1st ed.). Springer International Publishing.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Koza, J. R. (1992). *Genetic Programming : On the programming of computers by means of natural selection*. MIT Press.

Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, *194*(36), 3902–3933. https://doi.org/10.1016/j.cma.2004.09.007

Lera, D., & Sergeyev, Y. D. (2018). GOSH: derivative-free global optimization using multi-dimensional space-filling curves. *Journal of Global Optimization*, *71*(1), 193–211. https://doi.org/10.1007/s10898-017-0589-7

Liang, J. J., Suganthan, P. N., & Deb, K. (2005). Novel composition test functions for numerical global optimization. *Swarm Intelligence Symposium*, 68–75.

Liuzzi, G., Lucidi, S., & Piccialli, V. (2010). A partition-based global optimization algorithm. *Journal of Global Optimization*, *48*(1), 113–128. https://doi.org/10.1007/s10898-009-9515-y

Mantegna, R. N. (1994). Fast, accurate algorithm for numerical simulation of L\'evy stable stochastic processes. *Physical Review E*, *49*(5), 4677–4683. https://doi.org/10.1103/PhysRevE.49.4677

Mezura-Montes, E., & Coello, C. A. C. (2005). A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, *9*(1), 1–17. https://doi.org/10.1109/TEVC.2004.836819

Mezura-Montes, Efrén, & Coello, C. A. C. (2008). An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *International Journal of General Systems*, *37*(4), 443–473. https://doi.org/10.1080/03081070701303470

Mirjalili, S., Dong, J. S., & Lewis, A. (2020). *Nature-Inspired Optimizers; Theories, Literature Reviews and Applications*. Springer, Cham.

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, *114*, 163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*, *69*(Supplement C), 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Mirjalili, S., Saremi, S., Mirjalili, S. M., & Coelho, L. dos S. (2016). Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications*, *47*, 106–119. https://doi.org/10.1016/j.eswa.2015.10.039

Montes, E., & Ocana, B. (2008). Bacterial foraging for engineering design problems: preliminary results. *4th Mexican Congress on Evolutionary Computation (COMCEV'2008)*, 33–38. Mexico.

Parouha, R. P., & Das, K. N. (2016). A memory based differential evolution algorithm for unconstrained optimization. *Applied Soft Computing*, *38*(Supplement C), 501–517. https://doi.org/10.1016/j.asoc.2015.10.022

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., & Liew, K. M. (2003). Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation*, *7*(4), 386–396. https://doi.org/10.1109/TEVC.2003.814902

Reynolds, A. M., & Frye, M. A. (2007). Free-Flight Odor Tracking in Drosophila Is Consistent with an Optimal Intermittent Scale-Free Search. *PLOS ONE*, *2*(4), e354. https://doi.org/10.1371/journal.pone.0000354

Schluessel, V., & Bleckmann, H. (2012). Spatial learning and memory retention in the grey bamboo shark (Chiloscyllium griseum). *Zoology*, *115*(6), 346–353. https://doi.org/10.1016/j.zool.2012.05.001

Sergeyev, Ya 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), 1–9. https://doi.org/10.1038/s41598-017-18940-4

Sergeyev, Yaroslav D., & Kvasov, D. E. (2006). Global Search Based on Efficient Diagonal Partitions and a Set of Lipschitz Constants. *SIAM Journal on Optimization*, *16*(3), 910–937. https://doi.org/10.1137/040621132

Sims, D. W., Southall, E. J., Humphries, N. E., Hays, G. C., Bradshaw, C. J. A., Pitchford, J. W., … Metcalfe, J. D. (2008). Scaling laws of marine predator search behaviour. *Nature*, *451*(7182), 1098. https://doi.org/10.1038/nature06518

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

Tanabe, R., & Fukunaga, A. (2013). Success-history based parameter adaptation for Differential Evolution. *2013 IEEE Congress on Evolutionary Computation*, 71–78. https://doi.org/10.1109/CEC.2013.6557555

van den Bergh, F., & Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories. *Information Sciences*, *176*(8), 937–971. https://doi.org/10.1016/j.ins.2005.02.003

Viswanathan, G. M., Afanasyev, V., Buldyrev, S. V., Murphy, E. J., Prince, P. A., & Stanley, H. E. (1996). Lévy flight search patterns of wandering albatrosses. *Nature*, *381*(6581), 413. https://doi.org/10.1038/381413a0

Viswanathan, G. M., Buldyrev, S. V., Havlin, S., da Luz, M. G., Raposo, E. P., & Stanley, H. E. (1999). Optimizing the success of random searches. *Nature*, *401*(6756), 911–914. https://doi.org/10.1038/44831

Viswanathan, G. M., Raposo, E. P., & da Luz, M. G. E. (2008). Lévy flights and superdiffusion in the context of biological encounters and random searches. *Physics of Life Reviews*, *5*(3), 133–150. https://doi.org/10.1016/j.plrev.2008.03.002

Wetter, M. (2004). *Simulation-based building energy optimization* (PhD thesis). University of California : Berkeley.

Yang, X. (2010). *Engineering optimisation: an introduction with metaheuristic applications*. John Wiley and Sons.

Yang, X.-S., Cui, Z., Xiao, R., Gandomi, A. H., & karamanoglu, M. (2013). *Swarm Intelligence and Bio-Inspired Computation; Theory and Applications* (1st ed.). Elsevier.

Yang, X.-S., & Deb, S. (2009). Cuckoo search via Lévy flights. *Nature & Biologically Inspired Computing (NaBIC 2009)*, 210–214.

Yang, X.-S., & Deb, S. (2013). Multiobjective cuckoo search for design optimization. *Computers & Operations Research*, *40*(6), 1616–1624. https://doi.org/10.1016/j.cor.2011.09.026

Yapici, H., & Cetinkaya, N. (2019). A new meta-heuristic optimizer: Pathfinder algorithm. *Applied Soft Computing*, *78*, 545–568. https://doi.org/10.1016/j.asoc.2019.03.012

Zainuddin, M., Kiyofuji, H., Saitoh, K., & Saitoh, S.-I. (2006). Using multi-sensor satellite remote sensing and catch data to detect ocean hot spots for albacore (Thunnus alalunga) in the northwestern North Pacific. *Deep Sea Research Part II: Topical Studies in Oceanography*, *53*(3), 419–431. https://doi.org/10.1016/j.dsr2.2006.01.007

Zhang, J., Liang, C., Huang, Y., Wu, J., & Yang, S. (2009). An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization. *Applied Mathematics and Computation*, *211*(2), 392–416. https://doi.org/10.1016/j.amc.2009.01.048

Zhang, M., Luo, W., & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences*, *178*(15), 3043–3074. https://doi.org/10.1016/j.ins.2008.02.014

Zhang, Y. (2009). Parallel EnergyPlus and the development of a parametric analysis tool. *Proceedings of the 11th International IBPSA Conference*. Presented at the Scotland. Scotland.