Chapter 5

Murmuration-Flight based Dispersive Optimization Algorithm

Modern hybrids are in demand for current needs; thus, following the same spirit, new optimizers have been introduced in earlier chapters. These optimizers either combine the strength of pre-existing optimizers or interdisciplinary concepts to overcome the limitations of EC, as discussed in Chapter 2. Although they have the capability to address the issues, they also raise a level of conceptual complexity for the researchers involved in applied research other than optimization or ML. Thus, the crux of this chapter is to design a novel optimizer inspired by natural phenomena around us, resulting in a simple, easy-to-use, efficient algorithm in terms of time and search capability. For its broader applicability, it has been validated on benchmark functions and two challenging applications of AI as well.

5.1 Introduction

EC has emerged as one of the most sought-after techniques in diverse applications. The preeminent goal of any EC method is to efficiently explore and exploit the search space in order to produce high-quality solutions. A perfect balance of these two factors

is essential for achieving good results. When performing an optimization task, one of the most significant challenges that EC algorithms face is becoming stuck in local minima when a globally optimal solution is desired. Furthermore, NFL [225], states that there is no universally efficient optimization algorithm exists. There have been ongoing efforts among research communities to develop more efficient methods in terms of time and search space exploration. In earlier chapters, optimization techniques have been designed to overcome the limitations of ECs either by combining the pre-existing optimizer or interdisciplinary concepts. These optimizers are well-suited for different problems. Still, as a characteristic of any hybrid model, they persist the conceptual complexity in addition to other qualities like self-adaptive, stable performance, superior convergence, scalability, and solution diversity.

Moreover, several pre-existing optimizers suffer from scalability, slow convergence, local optima stagnation, and low solution diversity. They are also computationally expensive for advanced industrial applications such as big data analysis and other AI problems. Thus, the foremost objective of this study is to design a novel optimizer that is simple, and efficient in terms of time and search space exploration such that it is suitable for modern real-world applications. With similar inspiration, this chapter entirely dedicates itself to accomplishing these objectives by exploring natural phenomena around us. It has been observed that bird migration patterns are fascinating natural phenomena that can give a strong mathematical base model for new optimization methods.

Ornithologists use the term migration to signify a regular return movement of birds each year between separate breeding and wintering areas. In the process, birds can cover hundreds or thousands of kilometers, and some can cross inhospitable areas such as seas, deserts, or high mountain ranges. Migration is shown by many kinds of animals, including butterflies and other insects, mammals, marine turtles, and fish, but in none is it as extensively developed as in birds. The collective travel routes of birds span

almost the entire globe, with some extreme return journeys covering more than 30,000 km. As a result of migration, bird distributions are continually changing – in regular seasonal patterns and on local, regional, or global scales. Birds show various movements while migrating from one place to another. Migration in birds, in general, has been classified by scientists to be of five types – (i) routine day-to-day movements, (ii) one-way dispersal movements, (iii) dispersive migration involving post-breeding movements in any direction, (iv) irruptive or invasive migration and (v) nomadic migration [257]. While in flight, the group of birds also displays various formations ranging from the typical "V" formation to the "Starling Murmuration." Specific formations are essential to the birds in the navigating path. In the context of an optimization problem, the search space can be thought of as a three-dimensional space consisting of two main logical geographical points- the starting point (from where migration begins) and the endpoint (the optima, where the birds reach after migration). Here, we present an optimization technique based on the fascinating natural phenomena above.

"Evolution is a new revolution in AI". It leads the exponential growth of designing intelligent ML algorithm and their applications by utilizing the Darwin principles. Also, this combination seems like a future of AI where several tasks can be performed optimally without human intervention in the area of big data analysis, data privacy, robotics and trajectory optimization, etc. Big data analysis and data privacy are in current demand due to the emergence of Industry 4.0, resulting in smart digital services, automated diagnostic tools, and so on. These automated tools are well-equipped with intelligent learning algorithms for feature engineering, including feature extraction, feature selection, and classification models. In addition to several benefits, these digital smart applications include sensitive data transmission over communication channels and involve an additional risk of data theft. It is one of the most critical issues which may cause life threats to patients in the case of medical data because manipulating small content leads to the wrong diagnosis. Additionally, industrial data

usually have high-dimensional feature space, and ML techniques are applied for analysis. As we know, the presence of irrelevant, noisy, and redundant features in the data has been detrimental to the predictive performance of ML techniques. Therefore, an efficient optimization technique is needed, which can search the 2^n search space with n features. Further, it is also worth to be noted that several real-world applications have more than one conflicting objective and need to be optimized simultaneously. Classical optimization has limited ability, and ECs are preferable, increasing demands for efficient multi-objective optimization techniques.

Therefore, in the later part of this chapter, we address two challenging issues apart from introducing optimization techniques for single and multi-objective optimization problems. First is the optimal key generation for image encryption, and the other is optimal feature subset selection for classification using the proposed novel optimizer "Murmuration-Flight based Dispersive Optimization (MDO)". Results show that it efficiently finds optimal qualitative feature subsets in a lower timestamp without significantly compromising performance. Besides this, it is also effective in generating optimal keys for the encryption scheme. The same has been investigated and presented in this chapter.

5.2 Motivation and Contribution

Nature never ceases to fascinate with a rich abundance of awe-inspiring phenomena that it uses to sustain life. One such fascinating phenomenon called migration is the amazing way in which flocks of birds move from one place to another seasonally for breeding/food or other reasons. These little creatures seem to use some mysterious form of knowing exactly where they have to go and how. An inspiring feature of their flight is the coordination among the birds in a flock. Studies have observed that while migrating, birds usually follow a specific V-shaped pattern (Figure 5.1 a) in several species. In this V-shaped pattern, the most expert navigator bird holds a position at

the tip of the V. At the same time, relatively less experienced fliers occupy positions towards the back, thereby giving a better navigator at the front arrangement. This pattern provides them with flight stability and better coordination.

Another unique flight pattern called murmuration is displayed by some species like Starlings (Sturnidae). In a starling murmuration, a huge number of starlings flock together to form haphazard, large aerial shapes (Figure 5.1 b) while flying. Every starling in the flock communicates with its neighbors for navigation guidance. Moreover, flying in such shapes provides them warmth and protection from predators.

The concepts mentioned earlier can pose efficient optimization methods because migration is, in a way, nature's own way of optimizing the flight of birds. A population-based method can be modeled, which involves the selection of leaders that disperse in various directions, mimicking dispersive migration while also involving communication between the members for reaching optima. Also, the proposal of a new optimizer is supported by NFL. Aside from that, data privacy is a major concern of the smart healthcare industry, and encryption is one of the feasible solutions to the data security issue. Several works have employed encryption with a chaotic map. Encryption with chaotic maps, on the other hand, is extremely sensitive to initial conditions. As a result, if the input parameters are not chosen carefully, it can produce unacceptably low performance. EC method is an efficient technique to address such challenges. The key generation problem can be formulated as an optimization problem, which can be addressed using an optimization approach. Further, optimal feature subsets selection can also be formulated as an optimization problem. Compared to other techniques, EC-based selection gives superior results while considering the individual feature interaction property.

This chapter proposes a novel NIA based on three natural phenomena: (1) starling murmuration, (2) the flight pattern of migrating birds like the Canada goose (Branta canadensis), and (3) dispersive migration seen in some birds like the Atlantic puffin

(a) (b)

Figure 5.1: Representation of (a) V-shaped flight of migrating birds (b) Starling murmuration.

(Fratercula arctica). We also incorporate the Lévy flight behavior exhibited by certain species of animals and birds as it has proven to enhance the efficiency of randomization-based search algorithms. To the best of our knowledge, the proposed algorithm in this work is the first of its kind to utilize Lévy flights to initialize the first population of solutions, thereby ensuring better exploration of the search space from the starting point of the optimization process.

Apart from that, the major contributions of this study are summarized as follows:

1. We propose to incorporate the concept of starling murmuration for interaction among the solutions. It is assumed that every solution (or bird) interacts with its p'th neighbor. The value of p can be experimented with and chosen to be very small if the problem demands a better local search. Or, it can be chosen to be sufficiently large if the problem emphasizes the attainment of global optima.

2. We validated the performance of the proposed MDO algorithm against fifteen standard test functions of varying complexity and reported the best, average, and worst fitness values along with average execution time.

3. To verify the effectiveness of MDO, a comparative analysis with respect to pre-existed and popular nine optimizers on benchmark functions are presented. We

also performed the non-parametric test at a 5% significance level, and the statistics are reported.

4. Multi-objective MDO (MDO-M) algorithm is also proposed in this study to make it suitable for optimization problems consist more than one conflicting objective. It has been validated against standard test suits on different evaluation metrics.

5. Once its effectiveness is validated after that, to show the broader applicability of MDO, it has been applied to image encryption by providing an optimal key. The performance is evaluated using the evaluation metrics Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM). Further, the effect of selecting two different objective functions for an optimal key generation has been shown for image encryption in the spatial domain.

6. Both proposals have also been validated for optimal feature subset selection for ML classification problems. A vivid range of datasets with different features, instances, and classes have been utilized to get a comprehensive idea.

7. Through experimental analysis of benchmark functions, datasets classification, and real-world image encryption applications, it has been observed that the proposed method outperforms or yields competitive results.

The proposed algorithm has certain advantages with respect to other state-of-the-art methods. First, there are very few parameters, namely the population size and the value of p (the p'th neighbor of bird), that need to be modified to suit different domains of problems. This makes the proposed method relatively simple to implement and understand. Second, the proposed methods provide better exploration and exploitation due to incorporating Lévy flight-induced randomization at integral steps. Finally, the proposed method can be seamlessly applied to any real-world problem. Thus simplicity, efficiency, and better adaptability make the proposed methods strong contenders

for being considered for the purpose of optimization compared to other algorithms. Moreover, the candidateship is supported by the NFL Theorem.

5.3 Theoretical Background

This work is concentrated on a novel MDO and its applications to optimal key generation in image encryption and optimal feature subset selection for data classification. Therefore, in this section, some essential preliminaries have been discussed to understand the proposed methods better.

5.3.1 Randomization in Optimization

Randomization plays an integral role in the design of optimization algorithms. Better randomization is the key to efficient exploration and exploitation of the search space. Randomization can be achieved in a number of ways, the most common way being through random walks. Random walks are essentially a sequence of consecutive random steps taken over successive iterations in the search space. Random walks' randomization can be increased by choosing step sizes from specific distributions, such as the Lévy distribution, which has shown impressive results in this context over the years.

5.3.2 Lévy Flight

A random walk that uses Lévy distribution to draw its step sizes from is called Lévy flight. It is typically expressed in terms of a simple power-law formula $L(s) \sim |s|^{-1-\beta}$ where $0 < \beta \leq 2$ is an index [258]. The Lévy distribution can be mathematically described as

$$L(s,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \frac{e^{-\gamma/2(s-\mu)}}{(s-\mu)^{3/2}}, \text{ if } 0 < \mu < s < \infty \\ 0 \quad otherwise \end{cases} \tag{5.1}$$

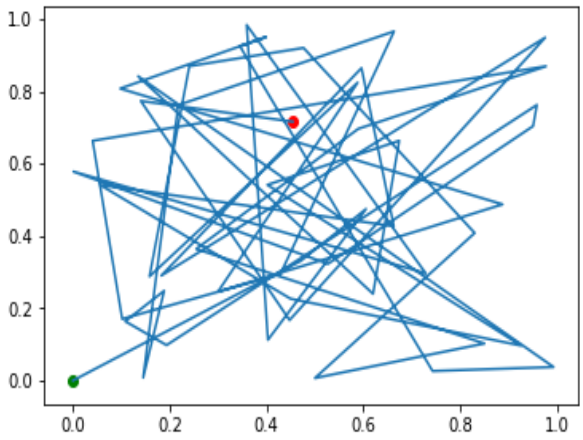


Figure 5.2: A sample Lévy flight.

where $\mu > 0$ is a minimum step and γ is a scale parameter. It must be noted that as $s \to \infty$, we have

$$L(s, \gamma, \mu) \approx \sqrt{\frac{\gamma}{2\pi}} \frac{1}{s^{3/2}} \tag{5.2}$$

Lévy flights are more efficient than the normal Brownian motion-based random walks in exploring unknown, large-scale search spaces. The reason for the same is attributed to the infinite mean and variance of the Lévy distribution that allows better global as well as local search space exploration. The implementation of Lévy flight consists of two major steps: (i) choosing a random direction, which is generally chosen from a uniform distribution, and (ii) generation of steps that obey Lévy distribution. One of the most efficient ways of implementing step (ii) is by using Mantegna's Algorithm (MA). According to MA [259], the step length (say step) can be calculated as

$$step = \frac{u}{|v|^{1/\beta}} \tag{5.3}$$

where u and v are drawn from Normal distribution, i.e.,$u \sim N(0, \sigma_u^2)$ and $v \sim N(0, \sigma_v^2)$ such that

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta) sin(\frac{\pi\beta}{2})}{\Gamma\left[\frac{1+\beta}{2}\right] \beta 2^{\frac{(\beta-1)}{2}}} \right\}^{\frac{1}{\beta}}, \sigma_v = 1 \tag{5.4}$$

Figure 5.2 shows a Lévy flight for 50 consecutive steps starting from (0,0). This version

of Lévy flight has been self-implemented to suit the problem in concern. The generation of new solutions x_i^{t+1} is performed using L*é*vy flights as

$$x_i^{t+1} = x_i^t + \alpha \oplus Lévy(\lambda) \tag{5.5}$$

L*é*vy distribution has an infinite variance and an infinite mean calculated as

$$Lévy \sim u = t - \lambda\,, \quad (1 < \lambda \leq 3) \tag{5.6}$$

Here, x_i^{t+1} and x_i^t represent the 'i' solutions at (t+1) and (t) times, respectively. ($\alpha > 0$) denotes the step size, which may vary according to the corresponding problem's scales, and usually, α is taken as $O(1)$. Another operator, $\oplus$, denotes the product, which is basically entry-wise multiplication. The effectiveness of random walk using Lévy flight resides in its capacity to explore the search space due to the fact that its step size is substantially larger in the long run.

5.4 Proposed Algorithm: Single-objective MDO

The proposed algorithm MDO draws inspiration from natural phenomena like starling murmuration, patterns of flight in migrating birds, and the idea of dispersive migration. The steps in the proposed algorithm are explained as follows:

5.4.1 Population Initialization

The MDO is a population-based method wherein the initial population is generated via L*évy* flights in (5.7). A base vector of dimension $D(0, 0, \ldots, 0)$ is fed in as input to the L*évy* flight function to generate N such candidate solutions each of dimension D. Each of these solutions represents a flying bird, and the whole set of solutions is analogous

to a flock of migrating birds.

$$\begin{aligned} b_i &= Lévy(base) \\ where, base &= [0, 0, 0..0]^D \end{aligned} \tag{5.7}$$

Here, b_i represents the *i'th* bird in the flock, and the base is an all 0 vector of dimension *D*. The *Lévy* function is defined as shown in (5.8). In subsequent iterations, the base vector is replaced by the *current best* solution or the *best captain*.

$$\begin{aligned} Lévy(base) &= s + step * randn \\ where, s &= Lb + (Ub - Lb) * rand \\ and\ step &= \frac{L}{100} * \frac{u}{|v|^{\frac{1}{\beta}}} * (s - base) \end{aligned} \tag{5.8}$$

Here, randn $\sim N(0, 1)$ is a random value drawn from the Normal distribution in the range [0,1], Lb and Ub are the lower and upper bounds, respectively, $rand \sim U(0, 1)$ is a random value drawn from the Uniform distribution in the range [0,1], L is the typical length-scale of the problem and u and v are drawn from Normal distribution, i.e., $u \sim N(0, \sigma_u^2)$ and $v \sim N(0, \sigma_v^2)$ in (5.4).

5.4.2 Position Updating via Murmuration

Just like in real-world migration, these birds need to start from their current position and move all the way to their destination, which in this case, is the optimal point(s). For the purpose of correct navigation through the search space, these birds communicate with one another. More specifically, each bird communicates with its p'th neighbor for correct navigation guidance, mimicking the phenomenon of a starling murmuration. If the fitness of bird i happens to be worse than that of its p'th neighbor; they exchange their positions so that the better navigator bird (better solution) is at the top (analogous to being at the front of the flock) in (5.9)-(5.10). Thereby, the birds in the flock arrange themselves in a "better navigator at the front arrangement," which is similar to the V-

shaped flight pattern exhibited by migrating birds.

$$x_i^{curr\_gen+1} = \begin{cases} x_{(i+p)\%N}^{(curr\_gen)}, & if f(x_{(i+p)\%N}) \, is \, better \, than \, f(x_i) \\ x_i^{curr\_gen}, & otherwise \end{cases} \tag{5.9}$$

$$x_{(i+p)\%N}^{(curr\_gen+1)} = \begin{cases} x_i^{curr\_gen}, & if \, f(x_{(i+p)\%N}) \, is \, better \, than \, f(x_i) \\ x_{(i+p)\%N}^{(curr\_gen)}, & otherwise \end{cases} \tag{5.10}$$

5.4.3 Captains Selection and Dispersion

Following this, m leaders or captains are selected from the flock at random as

$$\begin{gathered} Cap_j = b_k \\ where, j \in [0, m-1] \quad and \quad k = rand(0, N-1) \end{gathered} \tag{5.11}$$

Here, Cap_j represents the $j'th$ captain and b_k denotes the $k'th$ bird. These m captains are allowed to disperse in different directions (mimicking dispersive migration), and the potential of each captain is evaluated on the basis of a fitness function. The most promising captain (*bestcaptain*) is found out as

$$\begin{gathered} bestcaptain = Cap_z, \quad where, z \in [0, m-1] \\ s.t., Fitness(Cap_z) = (best)_{[j \in 0, m-1]} \{Fitness(Cap_j)\} \end{gathered} \tag{5.12}$$

5.4.4 Death and Nomadic Movement of Birds

Subsequently, a pre-decided specific fraction of the worst navigator birds is discarded (analogous to death owing to harsh weather conditions or nomadic behavior displayed by certain birds). These birds are then replaced by new solutions that are generated by feeding in the most promising captain (*bestcaptain*) as input to the Lévy flight function as

$$b_x = Lévy(bestcaptain) \tag{5.13}$$

Algorithm 5.1: Generates new candidate via Levy Flight using MA

Input: A d-dimensional candidate solution:

$best = [x_1, x_2, \cdots, x_d]$ that is the best in the previous generation's population,$lowerbound(l)$, $Upperbound(u)$

Output: A new $d-$dimensional candidate solution $[x_1, x_2, \cdots, x_d]^{new}$

1 Set Lower bound ($Lb \leftarrow l$) and Upper bound ($Ub \leftarrow u$)

2 Set *Lévy* exponent $\beta \leftarrow \frac{3}{2}$

3 Apply Mantegna's algorithm to get step length.

$step \leftarrow \frac{u}{|v|^{1/\beta}}$ // Use equation(5.3) and equation(5.4)

4 Calculate *stepsize*

$stepsize \leftarrow 0.01 * step *(step\text{-}best)$

5 Apply random walk as // use equation(5.8)

$s \leftarrow Lb + (Ub - Lb) * random,$

$s \leftarrow s + stepsize * rand\_normal$

6 Check for Lower bound and Upper bound consistency.

7 **return**: $[x_1, x_2, \cdots, x_d]^{new}$

Here, b_x represents one of the worse birds that have to be replaced. Finally, the whole flock is re-evaluated to get the new fitness values, and the best candidate bird in the current population is found out. This *current_best* shall be used as an input to the Lévy flight function in the following iteration (if any).

5.4.5 Termination

The whole process is carried out in successive iterations until a terminating criterion is reached, the terminating criterion being that the absolute difference between $current\_best fitness$ and $optimal\_fitness$ is less than or equal to a pre-defined threshold. Figure 5.3 depicts a flow diagram of the whole process.

The pseudo-code for the proposed MDO optimization algorithm is described in Algorithm 5.1 and Algorithm 5.2. Algorithm 5.1 describes the method to generate

Algorithm 5.2: MDO Algorithm

Input: Population Size:(N); fraction of birds to be discarded at each iteration:*(frac)*; each bird communicates with p^{th} neighbour:(p); number of captains:(m)

Output: Optimal solution concerned with the function or problem: $Best\_Fit$.

1 Generate initial population using **Algorithm 1**
$P = [p_1, p_2, \cdots, p_n]$; $best \leftarrow [0, 0 \cdots 0]^d$
2 **for** *a bird i* **do**
3 | **if** *Fitness*($P_{(i+p)\%N}$) *is better than Fitness*(P_i) **then**
4 | | $temp \leftarrow P_i$;
5 | | $P_i \leftarrow P_{(i+p)\%N}$;
6 | | $P_{(i+p)\%N} \leftarrow temp$
7 **end**
8 **repeat** m **times**
9 | $indx \leftarrow rand(0, N)$; // Choose m captains randomly.
10 | $Cap_i \leftarrow P_{indx}$
11 **end**
12 Find the best captain **begin**
13 | $j \leftarrow Min_j(Fitness([Cap_1, Cap_2, Cap_3, \cdots, Cap_m]))$
14 | $BestCap \leftarrow Cap_j$
15 **end**
16 Discard a fraction of the worst birds and replace them with new solutions.
$t \leftarrow frac * N$
17 **for** *k in range (N-t,N)* **do**
18 | $P_k \leftarrow$**Algorithm 1**$(Best\,Cap)$
19 **end**
20 Re-evaluate fitness and find Best_Fit
$z \leftarrow Min_z(Fitness(P))$;
$Best\_Fit \leftarrow P_z$
21 **if** $(Fitness(Best\_Fit) - Optimal\_Fitness) \leq 0.05$ **then**
22 | **return** $Best\_Fit$
23 **else**
24 | Go back to step 2
25 **end**

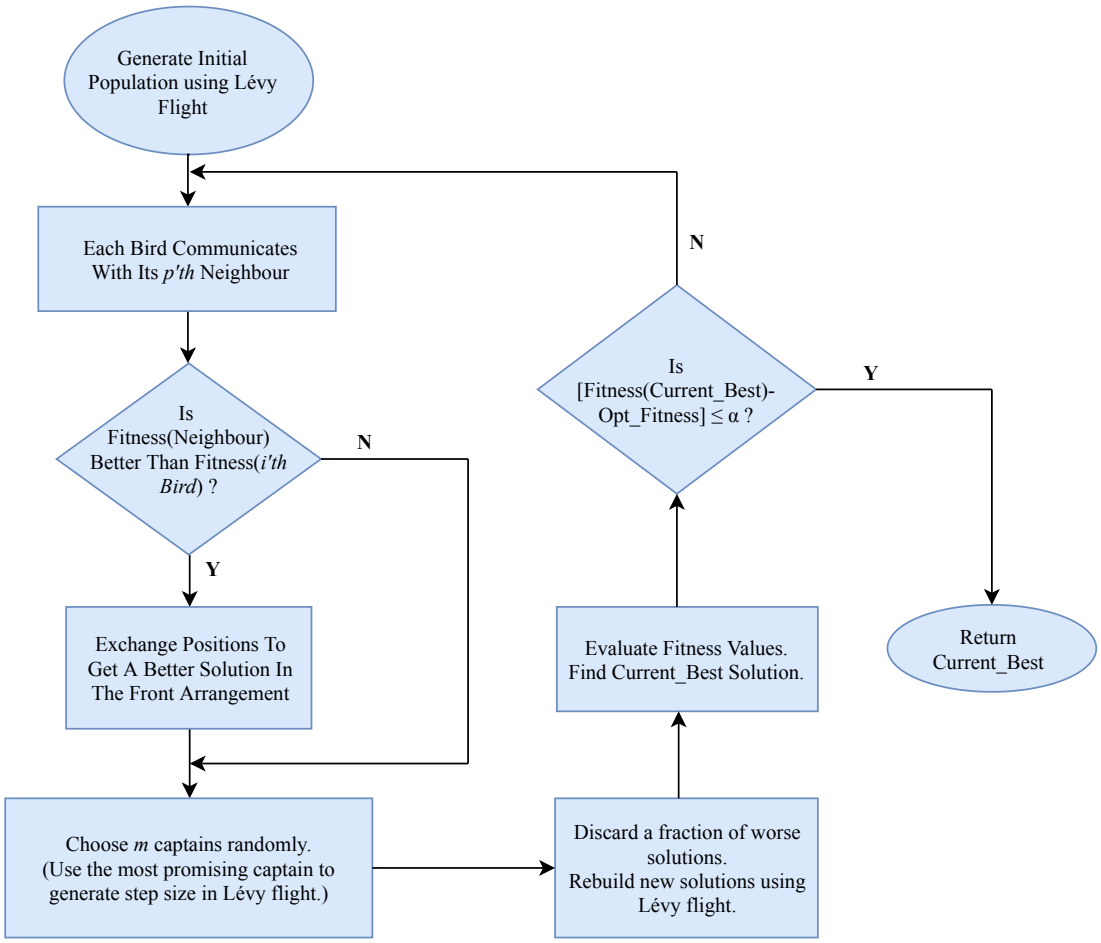


Figure 5.3: Flowchart of proposed MDO algorithm.

new solutions using Lévy flight, while Algorithm 5.2 describes the actually proposed algorithm.

5.5 Proposed Algorithm: Multi-objective MDO (MDO-M)

The proposed multi-objective optimization algorithm MDO is an extension of the earlier proposed MDO-S and utilizes the non-dominated sorting method along with crowding distance [208] for achieving the same. The ultimate goal of this multi-objective optimization algorithm, like others, is to find very accurate approximations of the true Pareto optimal solutions with the highest possible diversity.

5.5.1 Population Initialization

For the proposed multi-objective algorithm, a random population is initialized via Lévy flights in a fashion similar to that in its single-objective counter-part in (5.7) and (5.8).

5.5.2 Position Updating via Murmuration

Next, each bird interacts with its p'th neighbor so as to arrange themselves in a "better navigator at the front arrangement" on the basis of whether bird i dominates its p'th neighbor or otherwise.

$$x_i^{curr\_gen+1} = \begin{cases} x_{(i+p)\%N}^{(curr\_gen)}, \text{ if } x_{(i+p)\%N}) \prec x_i \\ x_i^{curr\_gen}, \quad otherwise \end{cases} \tag{5.14}$$

$$x_{(i+p)\%N}^{(curr\_gen+1)} = \begin{cases} x_i^{curr\_gen}, \text{ if } x_{(i+p)\%N}) \prec x_i \\ x_{(i+p)\%N}^{(curr\_gen)}, \quad otherwise \end{cases} \tag{5.15}$$

Here $a < b$ denotes solution a dominates solution b.

5.5.3 Captains Selection and Dispersion

Following this, non-dominated sorting is performed on the population, and the corresponding crowding distances are calculated. Only the non-dominant solutions (birds) are chosen as the m leaders/captains as

$$\begin{gathered} Cap_j = b_k,\ j \in [0, m-1] \quad where, k \in [0, N-1] \\ and \nexists\, p \in [0, N-1],\ p \neq k,\ s.t.,\ b_p \prec b_k \end{gathered} \tag{5.16}$$

Here, Cap_j represents the $j'th$ captain and b_k denotes the $k'th$ bird. Among these m captains, the most promising captain as best captain is selected by performing non-dominated sorting on the set of captains and selecting the first solution in the first level

of non-dominated solutions as

$$best\,captain = P_{opt_0^0} \tag{5.17}$$

Here, $P_{opt_i^j}$ represents the $i'th$ solution of the $j'th$ level of P_{opt}, the set consisting of all non-dominated solutions.

5.5.4 Death and Nomadic Movement of Birds

Now, all birds from the original population, barring the non-dominant birds, are discarded and replaced with new birds generated by inputting the best captain to the Lévy flight function as

$$b_x = Lévy(bestcaptain) \tag{5.18}$$

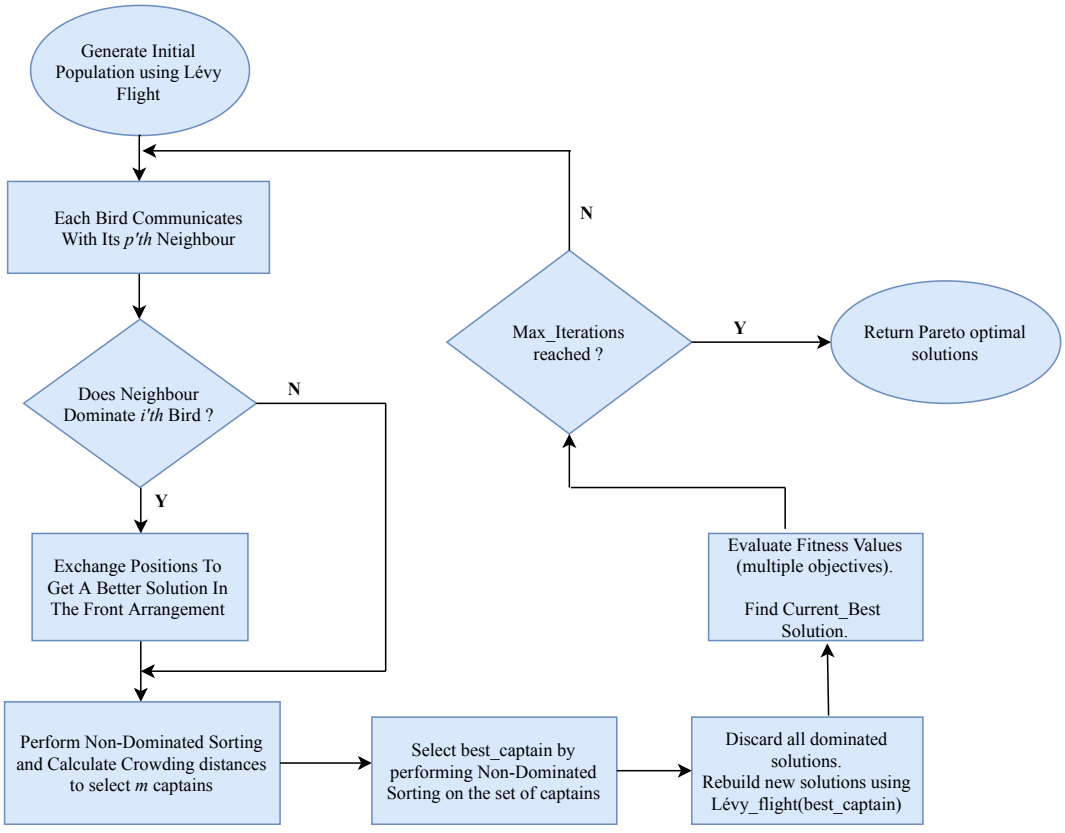


Figure 5.4: Proposed multi-objective optimization algorithm.

Algorithm 5.3: Multi-Objective MDO (MDO-M)

Input: Population size N, p (each bird communicates with its p'th neighbor)

Output: Pareto Optimal Solutions

1 Generate initial population using **Algorithm 1**
$P = [p_1, p_2, \cdots, p_n] = Algorithm1([0, 0 \cdots 0]^d)$
2 Iter $\leftarrow$ 0
3 $[Fit_1, ..., Fit_N] \leftarrow Objective_1(), ..., Objective_N()$
4 **for** *each bird i* **do**
5 | **if** $P_{(i+p)\%N} \prec P_i$ **then**
6 | | $temp \leftarrow P_i$;
7 | | $P_i \leftarrow P_{(i+p)\%N}$;
8 | | $P_{(i+p)\%N} \leftarrow temp$
9 **end**
10 Select m captains
$[Cap_1, Cap_2, ..., Cap_m] \leftarrow$ **Non_Dominated_Sort(P)**
11 Select best_captain
best_captain $\leftarrow$ **Non_Dominated_Sort**$([Cap_1, Cap_2, ..., Cap_m])$
12 **for** *each bird i:* **do**
13 | **if** *bird i* $\notin [Cap_1, Cap_2, ..., Cap_m]$: **then**
14 | | bird i $\leftarrow$ Algorithm 1(best_captain)
15 **end**
16 Re-evaluate Fitness Values
$[Fit_1, ..., Fit_N] \leftarrow Objective_1(), ..., Objective_N()$
17 Find Current_best $\leftarrow$ **Non_Dominated_Sort(P)**
18 Iter $\leftarrow$ Iter + 1
19 **if** $Iter == Max\_Iter$: **then**
20 | **return** Pareto Optimal Solutions
21 **else**
22 | Go back to step 3
23 **end**

Here, b_x represents one of the birds that does not belong to the non-dominated set of solutions and hence has to be replaced. All the fitness values (multiple objectives) are re-evaluated, and non-dominated sorting is performed one last time to find the *currentbest* solution/bird. This *currentbest* shall be used as an input to the *Lévy*

Algorithm 5.4: Non_Dominated_Sort Algorithm

Input: P=$[p_1, p_2, \cdots, p_N]$ where p_i=$[x_1, x_2, \cdots, x_D]$ D, A population of 'N' solutions each of dimension D

Output: F_{opt}, Optimal Fronts

```
for each p_i in P: do
    S_p ← {φ} ;   n_p ← {0}
    for each q_i in P: do
        if p_i ≺ q_i: then
            S_p ← S_p ∪ {q_i}
        else if p_i ≺ q_i : then
            n_p ← np + 1
    end
    if n_p == 0: then
        p_rank ← 1 ;   F_1 ← F_1 ∪ {p_i}
end
k ← 1
while F_k ≠ φ: do
    Q ← φ
    for each p in F_k: do
        for each q in S_p: do
            n_q ← n_q − 1
            if n_q == 0: then
                q_rank ← k + 1
                Q ← Q ∪ {q}
        end
    end
    k ← k + 1 ;   F_k ← Q
end
```

flight function in the following iteration (if any).

5.5.5 Termination

The process is repeated over successive iterations until a terminating condition (maximum number of iterations in this case) is reached. Finally, the first level of the non-

dominant solutions is returned as the Pareto optimal solutions. Figure 5.4 describes the flow diagram of the proposed MDO-M. The pseudo-code for the proposed MDO-M is described in Algorithms 5.3 and 5.4.

5.6 Application of MDO to Optimal Key Generation in Image Encryption

Data security is becoming increasingly relevant as open network communications are becoming more prevalent. Network information security is currently attracting a lot of interest from researchers all over the world. Image encryption has evolved into a method of transmitting digital image files with the greatest privacy and reliability. Currently, the healthcare system is integrated with IoTs, creating a smart healthcare system in which a large amount of data is transmitted over communication channels. Thus, medical image encryption plays a crucial role in smart healthcare applications. The medical image has received a lot of attention over the last decade, and a variety of methods [260, 261] have been developed to address the privacy concern. In addition, several alternative approaches [262] for optimal key generation in image encryption employing ECs have been developed. It is also worth noting that image encryption differs from text encryption due to some of the intrinsic qualities of images, such as large data capacity and strong data consistency; hence, traditional encryption methods are difficult to apply. As a result, we proposed a random optimum key-based encryption solution for images.

An optimization problem has been addressed in the suggested framework to generate the symmetric key for encryption. For encryption, AES [141] was used, which is based on a substitution-permutation network and is also a symmetric key symmetric block cipher approach. For this encryption, we attempt to generate a key that optimizes the visual quality of the encrypted image. Besides visual quality, we have also experimented with another fitness function that is formulated with the help of gap test [263].

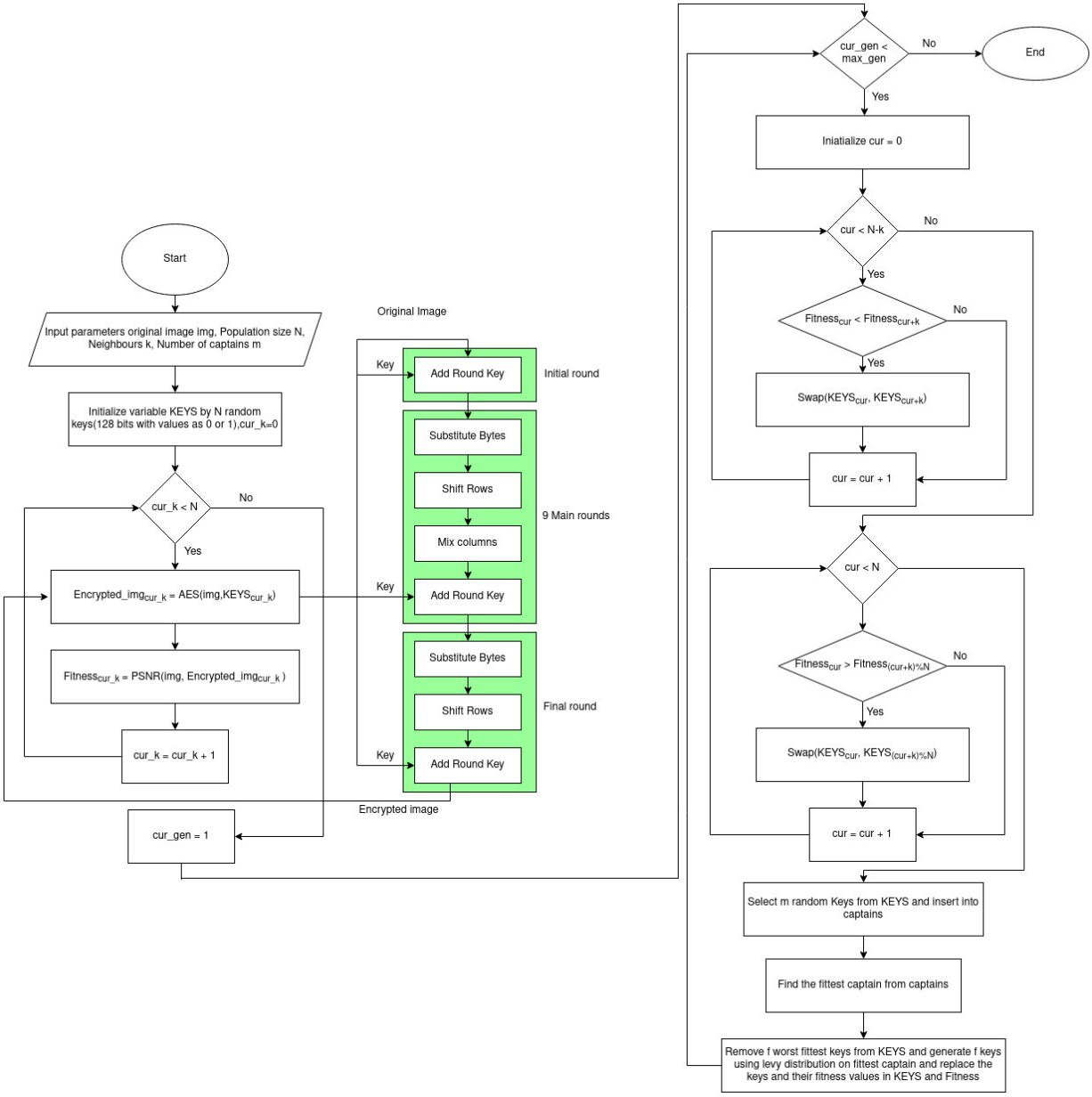


Figure 5.5: Flowchart for key optimization in image encryption.

The key can be generated using an optimization strategy, which has been described in Algorithm 5.5. For simplicity, the Algorithm 5.5 and flow chart presented in Figure 5.5 are explained using the visual metric as the objective function to be optimized. Before following the steps of the overall process, firstly, all the images are resized into 512×512 and converted to a grey scale.

Algorithm 5.5: MDO for Image Encryption

Input: Original image img, Population size N, Neighbours k, Captains m

Output: Key for the encryption and decryption K which maximizes PSNR value of the encrypted against Original image

Generate N random keys **Keys**

// Each key is of dimension 16 with each value in the range [0,255]

/* Start optimisation algorithm */

for $k \in$ **Keys do**

  Encrypted$_i \leftarrow$ AES(img, k) // Use AES for encrypting image with secret key as k

  fitness$_i \leftarrow$ PSNR(img, Encrypted$_i$)

end

cur_iteration $\leftarrow 1$

while *cur_iteration* $\leq$ *max_iteration* **do**

  for $i \leftarrow$ *1 to N-p* **do**

    if *fitness*$_i <$ *fitness*$_{i+k}$ **then**

      swap(**Keys**$_i$, **Keys**$_{i+k}$)

      swap(fitness$_i$, fitness$_{i+k}$)

    end

  end

  for $i \leftarrow$ *N-p+1 to N* **do**

    if *fitness*$_i >$ *fitness*$_{(i+k)\%N}$ **then**

      swap(**Keys**$_i$, **Keys**$_{(i+k)\%N}$)

      swap(fitness$_i$, fitness$_{(i+k)\%N}$)

    end

  end

  Captains $\leftarrow$ select m random solutions.

  Most-Fit $\leftarrow$ Captain with best fitness

  /* Remove f worst fit keys from **Keys** and replace them with new keys using Lévy distribution on most-fit */

  for $i \leftarrow$ *N-f+1 to N* **do**

    Keys$_i \leftarrow$ Lévy(Most-fit)

    Encrypted$_i \leftarrow$ AES(img, **Keys**$_i$)

    fitness$_i \leftarrow$ PSNR(img, Encrypted$_i$)

  end

end

return key with best fitness values from **Keys**

In brief, the overall process can be explained by following sub-steps.

1. Key initialization: A population of keys with random integer values in the range of [0,255] having key length 16 is initialized. In the successive generation, on the basis of the best fittest value, worst keys are replaced using Algorithm 5.1.

2. Fitness function computation: For this work, two different fitness functions, namely, PSNR and gap test, have been computed and showcased the impact of each on image encryptions. In the case of PSNR, each key has been used in the encryption algorithm for encrypting the original image, and thereafter $PSNR$ value is computed between the original and encrypted image. This value is assumed to be a fitness function, which is further optimized by MDO. Basically, $PSNR$ is an expression for the ratio of a signal's highest potential value to the power of distorting noise that influences the quality of its representation. The Mean Squared Error (MSE) and PSNR between two images of dimension N×M can be calculated by

$$MSE = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [img1_{ij} - img2_{ij}]^2 \tag{5.19}$$

$$PSNR = 10 \log_{10} \frac{MAX_{img1}^2}{MSE} \tag{5.20}$$

Here MAX_{img1} is the maximum signal value that exists in our original image, img1 is the original image, and img2 is the newly generated image.

It is clearly visible that the MSE helps us to equate the "true" pixel values of the original image to generated image under consideration. The MSE is measured as the mean of the squares of the errors between the real and corresponding generated images. This error is the measure of the difference in the images. So we need to maximize this error between the original and the encrypted images,

and thus would like to minimize the PSNR value as it is inversely proportional to the MSE. For encryption, we have used the AES 128-bit algorithm, which will be discussed in the next section.

We have also considered another fitness function gap test for key generation. Hence considered fitness functions$F_1(K), F_2(K)$ can be represented as an optimization problem for minimization separately as:

$$F_1(K) = \min(PSNR) \tag{5.21}$$

$$F_2(K) = \min(gaptest) \tag{5.22}$$

3. Fitness evaluation, selection, and optimal key generation: MDO is applied to optimize $F_1(K)$ and $F_2k)$ individually to generate an optimal key in each case. For simplicity, the whole process w.r.t fitness function shown in (5.21) is explained in Algorithm 5.5.

5.6.1 Encryption-decryption Scheme

AES-128 algorithm is used for image encryption. Also, AES is suitable for both interconnected hardware as well as software and is an efficient scheme for a cloud-based system. It is a symmetric block cipher scheme. This means that a similar key is used for encryption and decryption. The standard AES method has the limitation of not being a reliable security algorithm because the verification key is fixed and maintained in the database. Resulting, an unauthorized individual may gain access to the key. Hence, in the proposed method, by utilizing MDO, we tried to generate the automatic key for encryption that enhances the reliability of the AES scheme by restricting adversaries. AES-128 is operated on a plaintext of a fixed size (128 bits); therefore, we are using 128 bits keys for the algorithm. For 128-bit keys, AES consists of ten rounds, out of which all the rounds are identical except the last one. Since AES works on 16-byte blocks, we

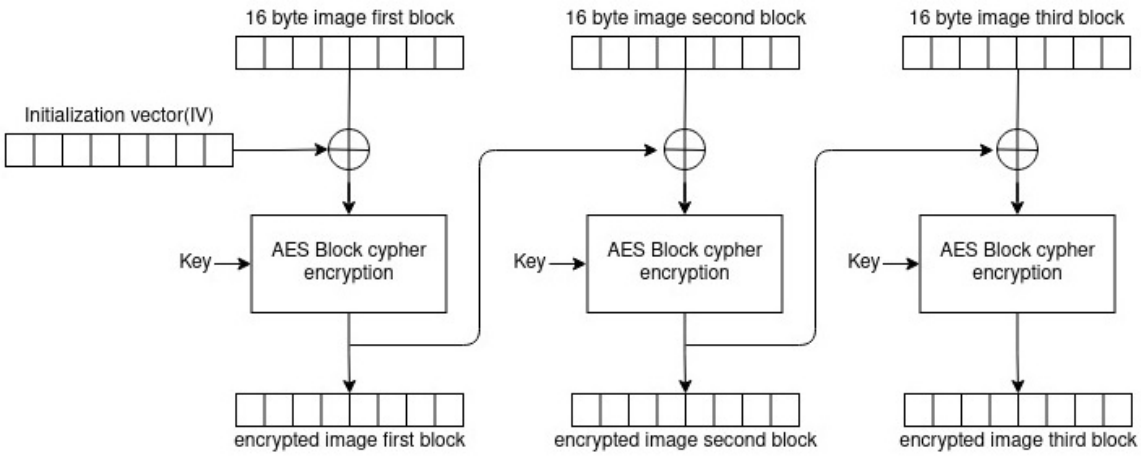


Figure 5.6: CBC mode in AES.

must first convert our image to 16-byte blocks. We have used the Cipher Block Chaining (CBC) mode of operation, where each block is XORed with the previous encrypted block before encryption. We can understand this concept by Figure 5.6.

The AES encryption for each block deals with 16 bytes blocks. These 16 blocks were then converted into 4×4 matrix as shown below:

$$Block = \begin{pmatrix} block_0 & block_4 & block_8 & block_{12} \\ block_1 & block_5 & block_9 & block_{13} \\ block_2 & block_6 & block_{10} & block_{14} \\ block_3 & block_7 & block_{11} & block_{15} \end{pmatrix} \tag{5.23}$$

The AES block cipher encryption consists of three main phases, as shown in Figure 5.5 and described below.

- Initial Round- Add Round Key.
- Main Rounds(9 Rounds) - Substitute bytes, Shift rows, Mix columns, and Add round key.
- Final Round - Substitute bytes, Shift rows, and Add round key.

The four operations used in the process are:

1. Add Round Key- This is the only operation that requires a key to be used. In this operation, the input is XORed with the key.

2. Substitute bytes- This operation includes the use of a 16×16 lookup table to replace all the bytes in the input matrix. This lookup table is known as the Substitution box (S-Box). For substitution, the 8-bit input is broken into two 4-bits. The first half will help in finding the row of the Box, and the second half will help in finding the column in the box. Using the row and column, we find a byte and replace the input byte with the byte found in the matrix.

3. Shift Rows- To jumble up the byte order in the block, we use this operation. In this operation, each of the rows in the matrix is shifted to the left by their row number. The top row (id=0) is not shifted since the row number is zero, the second row (id=1) is shifted by one, and so on, as can be seen below:

$$
\begin{aligned}
Input = & \begin{pmatrix} block_0 & block_4 & block_8 & block_{12} \\ block_1 & block_5 & block_9 & block_{13} \\ block_2 & block_6 & block_{10} & block_{14} \\ block_3 & block_7 & block_{11} & block_{15} \end{pmatrix} \\
\Longrightarrow & \begin{pmatrix} block_0 & block_4 & block_8 & block_{12} \\ block_5 & block_9 & block_{13} & block_1 \\ block_{10} & block_{14} & block_2 & block_6 \\ block_{15} & block_3 & block_7 & block_{11} \end{pmatrix}
\end{aligned} \tag{5.24}
$$

4. Mix Columns- It also mixes up the input as ShiftRows. It involves mixing up the bytes in each column separately. To be more specific, each byte in a column is replaced with 2× that byte, plus 3× the next byte, plus the byte that comes

next, plus the byte that follows.

$$Output = \begin{pmatrix} block_0 & block_4 & block_8 & block_{12} \\ block_1 & block_5 & block_9 & block_{13} \\ block_2 & block_6 & block_{10} & block_{14} \\ block_3 & block_7 & block_{11} & block_{15} \end{pmatrix} \times \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \tag{5.25}$$

Once the image is encrypted, the fitness values of each bird, i.e., the PSNR value between the encrypted and original image, have been computed. Thereafter, the main iterations for key optimization, shown in lines 7-17 of the Algorithm 5.5, are executed. Firstly, for each key, the bird communicates with its k^{th} preceding neighbor and swaps among them after comparing their fitness values. Keys associated with the worst fitness values have been removed from the original population. Further, captains are selected from these keys, and after that, the best fittest captain among the selected captains is chosen. Following that, the new keys are generated on the fittest captain using Algorithm 5.1 and used to replace the removed keys in the population. This process is repeated until the termination criteria is reached. At the end, the optimal key is generated as output, which is used for image encryption.

5.7 Application of MDO to Optimal Features Subset Selection

In this section, we describe the proposed method for feature selection. Feature selection is one of the core concepts in ML, which hugely impacts the performance of the model. Redundant features can negatively impact the overall efficiency of the model. Therefore, it is necessary to select the important and informative features out of the complete set of attributes. It is often a difficult task to decide which attributes to include and which ones to leave out [264]. This is where our proposed algorithm steps in. The proposed algorithm generates a set of solutions with dimensions equal to the number of features.

Consequently, the solution is converted from continuous to binary using (5.26). Let

Algorithm 5.6: Feature Selection Routine for MDO

Input: Input Dataset $D = \{(X_i, y_i)\}_{i=1}^{d}$
Output: Reduced Set of Optimal Features F^*

Generate an initial population using **Algorithm 1**
P=$[p_1, p_2, \cdots, p_n]$=Algorithm 1 $([0, 0 \cdots, 0]^d)$
Iter $\leftarrow 0$
$F^* \leftarrow \{\}$
$Fitness \leftarrow \{\}$
for *each* p_i *in P:* **do**
    S $\leftarrow \{\}$
    for *each* x_i *in* p_i*:* **do**
        if $x_i \geq 0.5$ **then**
            $S \leftarrow S \cup (X_i, y_i)$
    end
    $F^* \leftarrow F^* \cup \{S\}$
    Accuracy $\leftarrow$ Classifier(S)
    Fitness $\leftarrow Fitness \cup \{Accuracy\}$
end
for *each* p_i *in P:* **do**
    if *Fitness*$(P_{(i+p)\%N})$ *is better than Fitness*(P_i) **then**
        $temp \leftarrow P_i$;
        $P_i \leftarrow P_{(i+p)\%N}$
        $P_{(i+p)\%N} \leftarrow temp$
end
repeat m **times**
    $indx \leftarrow rand(0, N)$; // Choose m captains randomly
    $Cap_i \leftarrow P_{indx}$
end
Find the best Captain **begin**
    $j \leftarrow Min_j(Fitness([Cap_1, Cap_2, Cap_3, \cdots, Cap_m]))$
    $BestCap \leftarrow Cap_j$
end
$t \leftarrow frac * N$
for k *in range (N-t,N)* **do**
    $P_k \leftarrow$**Algorithm 1**$(Best\,Cap)$
end
Re-evaluate fitness and find Best_Fit
$z \leftarrow Min_z(Fitness(P))$
$Best\_Fit \leftarrow P_z$
$Best\_Fitness \leftarrow Fitness[z]$
$Best\_Features \leftarrow F^*[z]$
$Iter \leftarrow Iter + 1$
if *(Best_Fitness== Max_Accuarcy) or Iter == Max_Iter)* **then**
    return $Best\_Features$
else
    Go back to step 2
end

$[x_0, x_1, \cdots, x_n]$ be a solution in continuous space.

$$x_i' = \begin{cases} 1, \text{ if } x_i \geq 0.5 \\ 0, \quad otherwise \end{cases} \tag{5.26}$$

Corresponding to the binary solution that we have, we select those attributes from the original dataset that have a value of 1 against them. The new data set with a reduced set of features is then trained and tested against a classifier, and the performance metric is evaluated in terms of the obtained accuracy as

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{5.27}$$

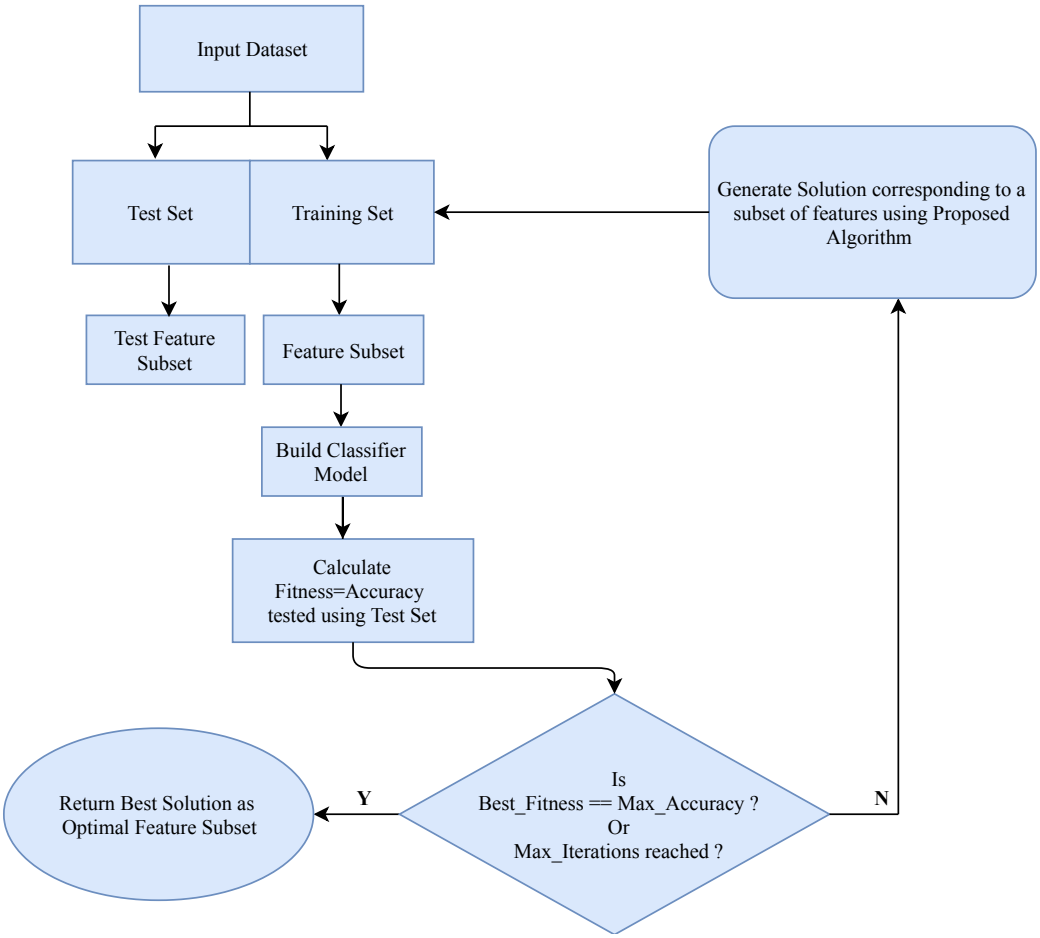


Figure 5.7: Feature selection using the proposed algorithm.

where TP, TN, FP, and FN are the True Positive, True Negative, False Positive, and False Negative values, respectively.

The above process is repeated for each solution in the population in successive iterations until the termination criterion is met. The termination criteria, in this case, are: (i) the *current_best* fitness (accuracy) is greater than or equal to a pre-set threshold value or (ii) the maximum number of iterations has been reached. Steps for the whole process are detailed in Algorithm 5.6, and Figure 5.7 describes the flow diagram of the above-explained process. It must be noted that the input dataset refers to a pre-processed (Min-Max normalized) dataset.

5.8 Results and Discussion

This section presents the experimental results for MDO and MDO-M on various benchmark test functions and test suites, respectively. In-depth analyses have also been provided for the best feature subset selection and the best key generation application. Both qualitative and quantitative analyses were done on the reported results. To assess the effectiveness and capabilities of proposals, a number of experiments were performed, and a comparative analysis with recently well-established optimizers was also studied. All experiments were performed in Python 3.7 on an Intel i7-8770 CPU, clocked at 3.20GHz with 6 cores and 16 GB RAM running Windows 10 as an operating system. The final parameters' values used in the algorithm are mentioned in Table 5.1.

5.8.1 Applications to Benchmark Test Problems

The novel MDO algorithm was used to optimize fifteen standard benchmarks, the specifications of which are detailed in Table 5.2. It must be noted that both unimodal (T6, T8, T10, T11, T12) and multi-modal (T1-T5, T7, T9, T13-T15) functions were used for the above-mentioned purpose. The corresponding parameters for the algorithm were decided empirically over a certain range of values for each parameter.

Table 5.1: Specifications of parameters used

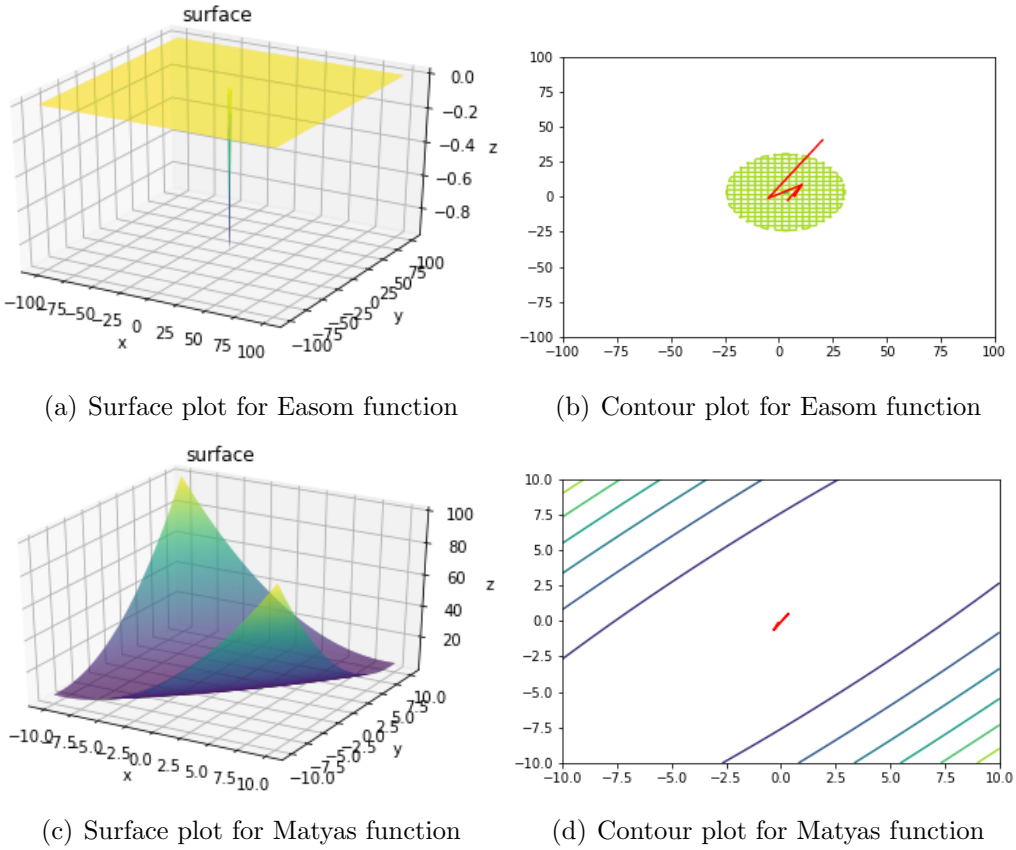
Parameters	**Value**
Population Size (N)	40
Fraction of birds discarded after each iteration	0.25
p (each bird communicates with its p'th neighbor)	7
m (number of captains/leaders)	20 (N/2)

Table 5.2 provides a detailed set of results with the best, worst and average fitness along with the average time (in seconds) obtained by the proposed algorithm against each of the test functions. The algorithm was run twenty times for each of the fifteen test functions, and the final fitness values, achieved after reaching convergence, were recorded. The standard deviation for each set of results corresponding to each test function was also noted. It can be inferred from Table 5.2 that the proposed algorithm is extremely accurate in achieving the optimal value of each test function, with the maximum error across all possible cases being less than 0.05. This can be attributed to a better search space exploration technique used by the proposed MDO by incorporating *Lévy* flights to generate initial solutions instead of using a mere random population as the initial population as well as better exploitation by selecting captains along with dispersive migration. The proposed algorithm is not dependent on generations and terminates only on reaching the convergence criterion, which has an error rate less than or equal to 0.05.

Figure 5.8 shows the surface plots of two of these test functions, namely, Easom (T6) and Matyas (T11), both of which happen to be unimodal functions, along with the corresponding contour plots showing the steps in reaching convergence by the proposed algorithm. Figure 5.9, on the other hand, shows the surface and contour plots for two multi-modal functions, namely, Shubert and Rosenbrock functions. From both Figure 5.8 and Figure 5.9, it can be seen that the proposed algorithm reaches the optima with good efficiency. The contour plot for the Easom function shows that the algorithm

Table 5.2: Performance of the proposed MDO algorithm on benchmark functions

Test Function		Equation/Formula	Range	Optimal Value	Best	Worst	Mean	Std.	Average Execution Time (s)
T1	Beale's	$(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.625-x_1+x_1x_2^3)^2$	[-4.5,4.5]	0	0.0025	0.04	0.021	0.014	0.08025710
T2	Egg holder	$-(x_2+47)sin(\sqrt{\lvert x_2+\frac{x_1}{2}+47\rvert})-x_1sin(\sqrt{\lvert x_1-(x_2+47)\rvert})$	[-512,512]	-959.64	-959.6406	-959.5936	-959.6265	0.015	46.4001428
T3	Bird function	$sin(x_1)\,e^{(1-cos(x_2))^2}+cos(x_2)\,e^{(1-sin(x_1))^2}+(x_1-x_2)^2$	[-2π, 2π]	-106.76	-106.7624	-106.7169	-106.7423	0.016	5.74895138
T4	Shubert	$\left(\sum_{i=1}^{5} i\,cos\left((i+1)\,x_1+i\right)\right)\left(\sum_{i=1}^{5} i\,cos\left((i+1)\,x_2+i\right)\right)$	[-10,10]	-186.73	-186.7281	-186.6814	-186.7054	0.014	32.4774980
T5	Michalewicz	$-\sum_{i=1}^{d} sin(x_i)\,sin^{2m}\left(\frac{i\,x_i^2}{\pi}\right),\ m=10$	[0, π]	-1.8013	-1.7968	-1.7531	-1.7771	0.013	0.19590871
T6	Easom	$-cos(x_1)\,cos(x_2)\,exp^{\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)}$	[-100,100]	-1.0	-0.9967	-0.9510	-0.9746	0.014	47.9365020
T7	Schwefel	$418.9829\,d-\sum_{i=1}^{d} x_i\,sin(\sqrt{\lvert x_i\rvert})$	[-500,500]	0	0.0019	0.0497	0.0233	0.015	70.574766
T8	Booth	$(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	[-10,10]	0	0.0044	0.0468	0.0218	0.014	0.51533477
T9	Rosenbrock	$\sum_{i=1}^{d}\left[100\,(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$	[-5,10]	0	0.0022	0.0472	0.0294	0.015	2.03657125
T10	Sphere	$\sum_{i=1}^{d} x_i^2$	[-5.12,5.12]	0	0.0052	1.0831	0.033	0.074	0.51927706
T11	Matyas	$0.26\,(x_1^2+x_2^2)-0.48x_1x_2$	[-10,10]	0	0.0002	0.0047	0.0025	0.001	0.23089028
T12	Schaffer N.2	$0.5+\frac{sin^2\left(x_1^2-x_2^2\right)-0.5}{\left[1+0.001\left(x_1^2+x_2^2\right)\right]^2}$	[-100,100]	0	0.051	0.4999	0.1329	0.091	4.56937917
T13	Periodic	$1+sin^2(x_1)+sin^2(x_2)-0.1e^{-\left(x_1^2+x_2^2\right)}$	[-10,10]	0.9	0.900	0.9453	0.9128	0.012	0.00446420
T14	Goldstein-Price	$[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]*[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	[-2,2]	3	3.001	3.049	3.023	0.018	4.97506300
T15	Three-Hump Camel	$2x_1^2-1.05x_1^4+\frac{x_1^6}{6}+x_1x_2+x_2^2$	[-5,5]	0	0.001	0.048	0.024	0.015	0.07466209



(a) Surface plot for Easom function

(b) Contour plot for Easom function

(c) Surface plot for Matyas function

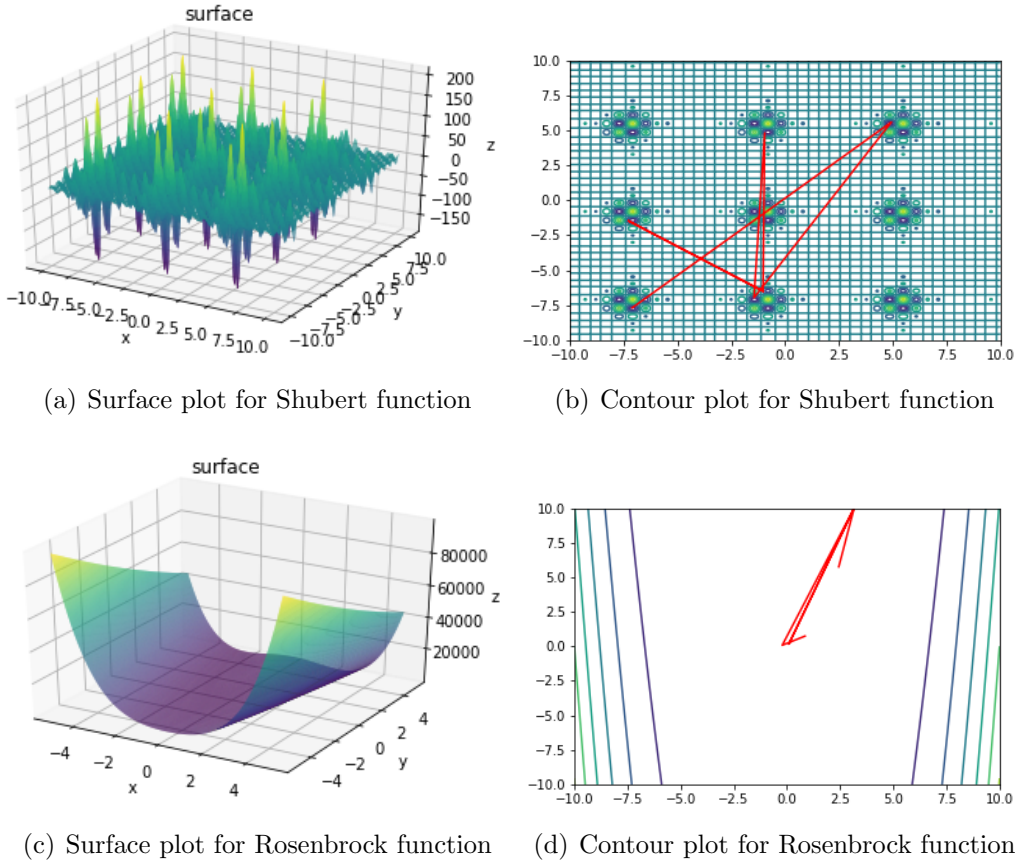
(d) Contour plot for Matyas function

Figure 5.8: Surface plot and contour plot for Easom function and Matyas function.

reaches the global minimum at (3.14,3.14) in Figure 5.8(b) while that for the Matyas function also reaches the global minimum at (0,0) in Figure 5.8(d). In the case of the multi-modal functions, the contour plot for the Shubert function in Figure 5.9(b) shows the algorithm going through the multiple global optima (the Shubert function has 18 global minima). The last contour plot for the Rosenbrock function shows that the algorithm reaches the global best at (1, . . . ,1) in Figure 5.9(d).

5.8.1.1 Comparative Analysis

The results of the proposed algorithm against these fifteen test functions were compared with that of nine other pre-established NIAs viz., PSO, firefly algorithm (FFA), GWO, GSA, cuckoo search (CS), multi verse optimizer (MVO)[54], salp-swarm algorithm (SSA) [56], whale optimization algorithm (WOA) [53], moth flame optimization



(a) Surface plot for Shubert function

(b) Contour plot for Shubert function

(c) Surface plot for Rosenbrock function

(d) Contour plot for Rosenbrock function

Figure 5.9: Surface plot and contour plot for Shubert function and Rosenbrock function.

(MFO) [265]. Since it is well known that different NIAs may favor different population sizes. Additionally, the optimal parameters for different test functions may vary. Therefore, it is quite hard to set a single parameter that is suitable for all algorithms and benchmarks. For the purpose of unbiased comparison, each algorithm was run for 500 iterations, with the population size in each case being initialized to 40. The observations were recorded and reported as tabulated in Table 5.3, describing the best, average, and worst fitness values attained by each algorithm against each test function. It can be seen from Table 5.3 that the proposed algorithm performs on par or better in all the test functions with almost all the other algorithms under consideration. The same was expected because the proposed MDO utilizes the combined approach of good initialization and better intra-population interaction to reach convergence.

Table 5.3: Comparison of the proposed algorithm with other well-established algorithms

Test Function		Proposed	PSO	MVO	FFA	SSA	WOA	GSA	GWO	MFO	CS
	Best	0.0001	0.0	5.31e-9	7.1e-11	4.5e-18	7.9e-15	1.9e-21	1.02e-8	0.0	1.0e-21
T1	**Avg**	0.011	0.0	0.072	9.8e-10	3.0e-15	0.072	4.5e-20	7.6e-8	1.5e-21	5.8e-15
	Worst	0.043	0.0	0.762	2.9e-9	1.8e-14	0.762	1.8e-19	2.07e-7	3.2e-10	1.2e-13
	Best	-959.64	-959.64	-959.64	-2911.1	-959.64	-959.64	-952.39	-959.64	-959.64	-2313.1
T2	**Avg**	-959.62	-824.88	-895.37	-3912.7	-956.54	-956.14	-713.49	-878.64	-955.11	-2629.7
	Worst	-959.59	-629.63	-718.16	-5001.1	-894.57	-888.94	-530.05	-716.66	-888.95	-2885.7
	Best	-106.76	-106.76	-106.76	-106.76	-106.76	-106.76	-106.76	-106.76	-106.76	-106.76
T3	**Avg**	-106.74	-106.76	-105.83	-106.76	-106.76	-106.76	-106.76	-105.83	-106.76	-106.76
	Worst	-106.71	-106.76	-87.31	-106.76	-106.76	-106.76	-106.76	-87.31	-106.76	-106.76
	Best	-186.72	-186.73	-186.73	-186.73	-186.73	-186.73	-186.73	-186.73	-186.73	-186.73
T4	**Avg**	-182.70	-186.73	-186.73	-186.73	-186.73	-186.73	-183.73	-186.72	-186.73	-186.73
	Worst	-180.68	-186.73	-186.72	-186.73	-186.73	-186.73	-172.61	-186.54	-186.73	-186.73
	Best	-1.00	-1.0	-0.99	-1.96	-1.0	-1.0	-1.0	-0.99	-1.0	-1.96
T5	**Avg**	-0.999	-1.0	-0.99	-1.83	-0.99	-0.99	-0.95	-0.99	-1.0	-1.95
	Worst	-0.994	-1.0	-0.99	-0.99	-0.99	-0.99	-0.41	-0.99	-1.0	-1.94
	Best	-0.677	-1.0	-0.99	-0.99	-0.99	-0.99	-1.0	-0.99	-1.0	-1.0
T6	**Avg**	-0.144	-1.0	-0.95	-0.81	-0.99	-0.95	-0.91	-0.99	-1.0	-0.99
	Worst	-6.604	-1.0	0.0	0.0	-0.99	0.0	-4.9e-5	-0.99	-1.0	-0.99
	Best	0.32	2.5e-5	2.63e-5	-2052.8	2.54e-5	3.35e-5	12.52	4.2e-15	2.54e-5	-1307.7
T7	**Avg**	7.77	44.17	73.31	-2853.7	2.54e-5	11.28	213.16	45.12	22.56	-1702.4
	Worst	39.23	236.87	236.87	-3849.1	2.54e-5	118.76	368.71	236.87	118.43	-1776.5
	Best	0.0	0.0	1.67e-7	3.9e-10	7.2e-17	3.53e-5	2.9e-21	1.51e-8	0.0	1.0e-26
T8	**Avg**	0.005	0.0	7.14e-7	3.7e-9	2.9e-14	0.0005	6.6e-20	4.21e-7	0.0	1.4e-23
	Worst	0.07	0.0	1.94e-6	1.37e-8	1.1e-13	0.002	3.3e-19	1.289-5	0.0	1.8e-22
	Best	0.0	0.0	5.97e-8	4.3e-11	7.3e-16	1.1e-11	0.004	1.5e-28	2.37e-9	4.3e-14
T9	**Avg**	0.08	3.43e-14	1.2e-6	1.24e-8	8.0e-14	1.2e-6	0.03	9.3e-7	0.004	4.3e-10
	Worst	0.18	5.67e-13	4.9e-6	7.86e-8	3.6e-13	5.5e-6	0.09	2.34e-6	0.032	8.45e-9
	Best	0.0	4.24e-71	6.1e-9	1.5e-11	6.2e-17	0.0	7.5e-24	0.0	0.0	1.9e-28
T10	**Avg**	0.008	3.93e-63	6.05e-8	6.2e-10	3.4e-15	0.0	1.6e-20	0.0	0.0	5.0e-25
	Worst	0.03	8.24e-62	2.02e-7	1.83e-9	1.8e-14	0.0	5.6e-20	2.54e-8	0.0	3.6e-24
	Best	0.0	5.08e-55	1.8e-10	7.4e-12	1.1e-15	0.0	1.2e-22	0.0	0.0	2.1e-28
T11	**Avg**	0.002	1.51e-47	2.73e-8	1.9e-10	1.3e-15	0.0	1.7e-21	0.0	0.0	7.3e-25
	Worst	0.008	3.09e-46	6.76e-8	8.1e-10	8.3e-15	0.0	6.5e-21	0.0	0.0	5.9e-24
	Best	0.0	0.0	5.9e-9	2.3e-10	1.1e-15	0.0	0.0	0.0	0.0	0.0
T12	**Avg**	0.07	0.0	1.48e-6	7.07e-9	0.0005	0.0	0.11	0.0	0.0	1.2e-15
	Worst	0.12	0.0	4.8e-6	3.59e-8	0.01	0.0	0.44	0.0	0.0	9.7e-15
	Best	0.9	0.9	0.9	0.9	0.9	0.9	0.91	0.9	0.9	0.9
T13	**Avg**	0.94	0.919	0.95	0.94	0.91	0.91	0.98	0.91	0.95	0.9
	Worst	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9
	Best	3.0	1.86e-14	3.0	3.0	2.99	3.0	2.99	3.0	2.99	2.99
T14	**Avg**	3.04	2.99e-15	3.0	3.0	3.0	3.0	2.99	3.0	2.99	2.99
	Worst	3.06	4.51e-18	3.0	3.0	3.0	3.0	2.99	3.0	2.99	2.99
	Best	6.7e-5	7.96e-15	6.1e-10	7.7e-11	9.1e-17	0.0	4.0e-21	0.0	0.0	1.8e-24
T15	**Avg**	0.009	0.072	5.0e-8	7.8e-10	1.7e-15	0.0	0.123	0.0	0.0	1.7e-22
	Worst	0.02	0.762	2.19e-7	2.2e-9	5.4e-15	0.0	0.773	0.0	0.0	1.3e-21

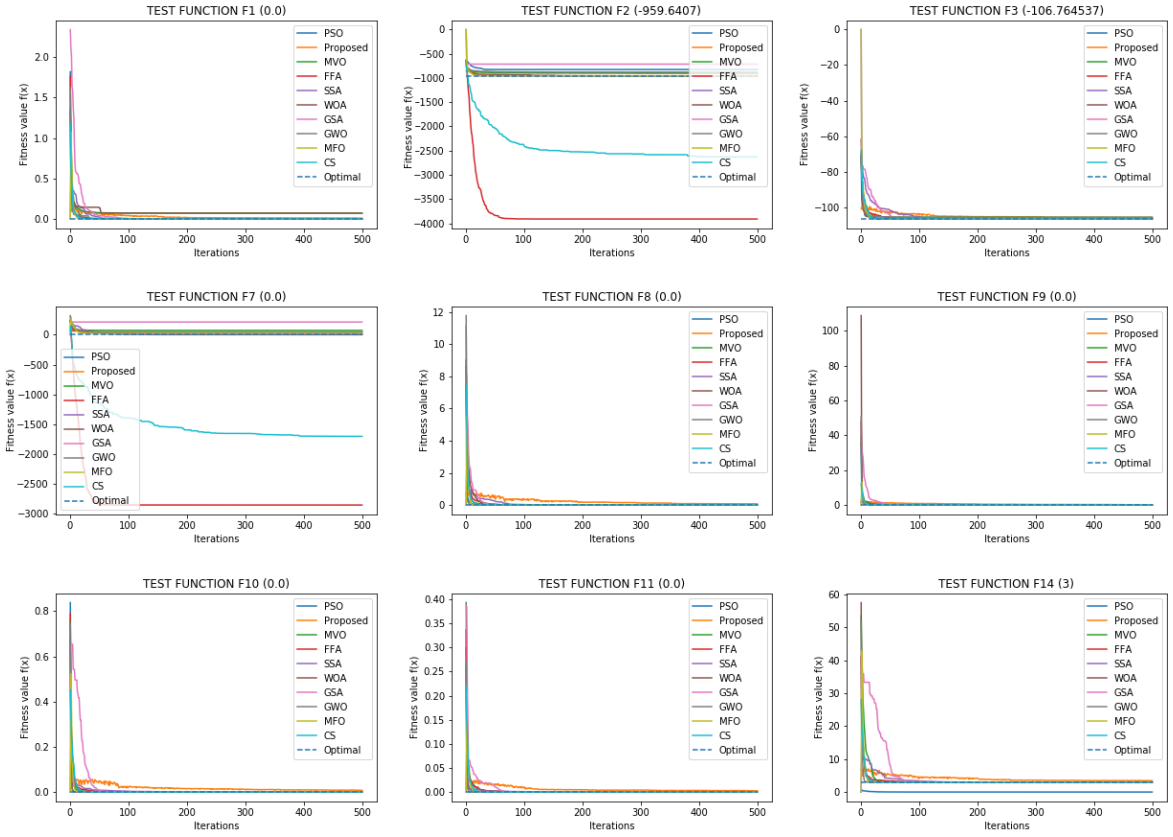


Figure 5.10: Comparison of proposed MDO with other EC methods

5.8.1.2 Experimental Analysis

To visualize the results attained in the above-mentioned experiments, convergence rate analysis is performed and some of which are depicted in Figure 5.10. A notable observation from the plots is that the proposed algorithm always starts the search (in the search space) from a position much closer to the global optima as compared to other algorithms. This better initialization strategy, in turn, renders the proposed algorithm more efficient in terms of search space exploration and computation time. It can also be seen from the last column of Table 5.2, which shows the average execution time of the proposed algorithm for each test function.

5.8.1.3 Non-parametric Test on Benchmark Functions

In order to further validate the results of our proposed algorithm, the non-parametric statistical Wilcoxon's rank sum test was performed based on the results of the proposed

Table 5.4: Results of Wilcoxon's Rank Sum test (at 5% significance level) with MDO against other optimization algorithms on fifteen benchmark functions

Test Function	Proposed-PSO		Proposed - MVO		Proposed – GWO		Proposed – MFO		Proposed – CS		Proposed – WOA		Proposed – FFA		Proposed – SSA		Proposed – GSA	
	P-value	Win	P-value	Win	P-value	Win	P-value	Win	P-value	Win	P-value	Win	P-value	Win	P-value	Win	P-value	Win
T1	0.000902	+	5.956 e-05	+	0.007838	+	5.956 e-05	+	5.956 e-05	+	5.956e-05	+	0.032549	+	0.000902	+	0.9307556	-
T2	0.042019	+	0.821259	-	0.011737	+	9.221 e-05	+	5.956 e-05	+	0.003704	+	0.000281	+	0.000162	+	0.000186	+
T3	5.956e-05	+	0.000367	+	0.001657	+	0.000214	+	5.956 e-05	+	0.000367	+	0.006362	+	0.006362	+	0.715144	-
T4	5.956e-05	+	0.001021	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	0.821259	-
T5	5.956e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	0.006362	+
T6	5.956e-05	+	9.221 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	6.898751	-	0.000477	+	5.956 e-05	+	5.956 e-05	+
T7	0.394457	-	0.098741	-	0.767657	-	0.0005434	+	5.956 e-05	+	0.414040	-	0.000186	+	0.1219318	-	0.001021	+
T8	5.956e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	0.000141	+	5.956 e-05	+	5.956 e-05	+	0.001304	+
T9	5.956e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	0.001021	+	5.956 e-05	+	0.000245	+
T10	5.956e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	7.980 e-05	+
T11	5.956e-05	+	5.956e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+
T12	5.956e-05	+	5.956e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	0.000245	+	5.956 e-05	+	0.000122	+
T13	0.0005434	+	0.032549	+	0.0014711	+	0.0011546	+	5.956 e-05	+	0.000214	+	0.000367	+	0.000245	+	0.035479	+
T14	5.956e-05	+	5.956e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	0.007065	+	5.956 e-05	+	5.956 e-05	+	0.042019	+
T15	5.956e-05	+	5.956e-05	+	5.956 e-05	+	5.956 e-05	+	5.956 e-05	+	0.098741	-	5.956 e-05	+	5.956 e-05	+	0.098741	-

algorithm against each of the other optimization algorithms at a 5% significance level. Results obtained from the test are summarized in Table 5.4 where the '+' sign indicates the reference algorithm MDO performed better than the compared algorithm, and the '−' sign indicates that the reference algorithm is inferior to the compared one. It can clearly be seen that the proposed algorithm is significantly better than MFO, CS, and FFA on all 15 test functions. It is significantly better than PSO, MVO, GWO, and SSA on 14 of the fifteen test functions, better than WOA on 12 test functions, and better than GSA on 11 test functions. It confirms that MDO demonstrates statistically significant and efficient than the others, which are considered for comparative analysis in Wilcoxon's rank sum test at $\alpha = 0.05$.

5.8.2 Performance Analysis of Proposed Encryption Scheme for Images in Spatial Domain

Following the establishment of the proposed algorithm against well-known test functions and a successful validation against nine other pre-established optimization algorithms, our proposed algorithm was put to examination on the problem of optimal key generation for image encryption. For this, two different fitness functions, namely PSNR and gap test, have also been considered, and their impact on the overall encryption mechanism has been shown on different images. A diverse set of publicly available images of size 512×512 is considered, including three medical images of various radiology, namely CT Scan Covid, MRI Brain, and Xray Chest, as well as two widely used natural images of an airplane and a boy. All of the medical images were collected from the image database; namely, OPENi [266].

We used the proposed MDO-based encryption method for these images, using two distinct fitness functions (PSNR and GAP Test) for optimal key generation, and observed that all encrypted images were successfully recovered with no visual impairment.

The value of parameters used by MDO for optimal key generation are provided in

Table 5.5: Parameters used for key generation using MDO

Parameter	Value	Parameter	Value
Population Size	20	Neighbour to communicate	7
Number of generation	5	Number of captains selected	5
Fraction of birds to be removed in each generation	0.25	β for Lévy distribution	1.5

Table 5.6: Statistics of evaluation metrics for encryption scheme

Image	PSNR			GAP Test		
	MSE↑	PSNR↓	SSIM↓	MSE↑	PSNR↓	SSIM↓
Airplane	10332.61328	7.98870	0.00055	10310.33594	7.99808	0.00039
Boy	10130.18750	8.07463	0.00015	10110.34473	8.08314	0.00003
CT-Covid	11482.54883	7.53042	0.00020	11472.98340	7.53404	0.00023
MRI-Brain	13613.24219	6.79119	0.00005	13605.72168	6.79359	0.00008
Xray-Chest	8034.60547	9.08116	0.00008	8012.24561	9.09326	0.00023

Table 5.5.

Further, we have evaluated MDO based encryption scheme by using the quality metrics (MSE, PSNR, SSIM) between the encrypted image and the original image. The original image, as well as the encrypted image along with their corresponding histogram, are shown in Figures 5.11–5.15. Correlation diagrams between two adjacent pixels (horizontal, vertical, and diagonal) of original images have been presented in Figure 5.11 (d), Figure 5.12 (d), Figure 5.13 (d), Figure 5.14 (d), and Figure 5.15 (d), which show that adjacent pixels in original images are strongly correlated. However, correlations of adjacent pixels (horizontal, vertical, and diagonal) of encrypted images are shown in Figure 5.11 (e), Figure 5.12 (e), Figure 5.13 (e), Figure 5.14 (e), and Figure 5.15 (e). It has been concluded from these representative images that the correlation of adjacent pixels in encrypted images is too low or negligible. It favors the requirement of privacy such that the adversary is unable to retrieve the information.

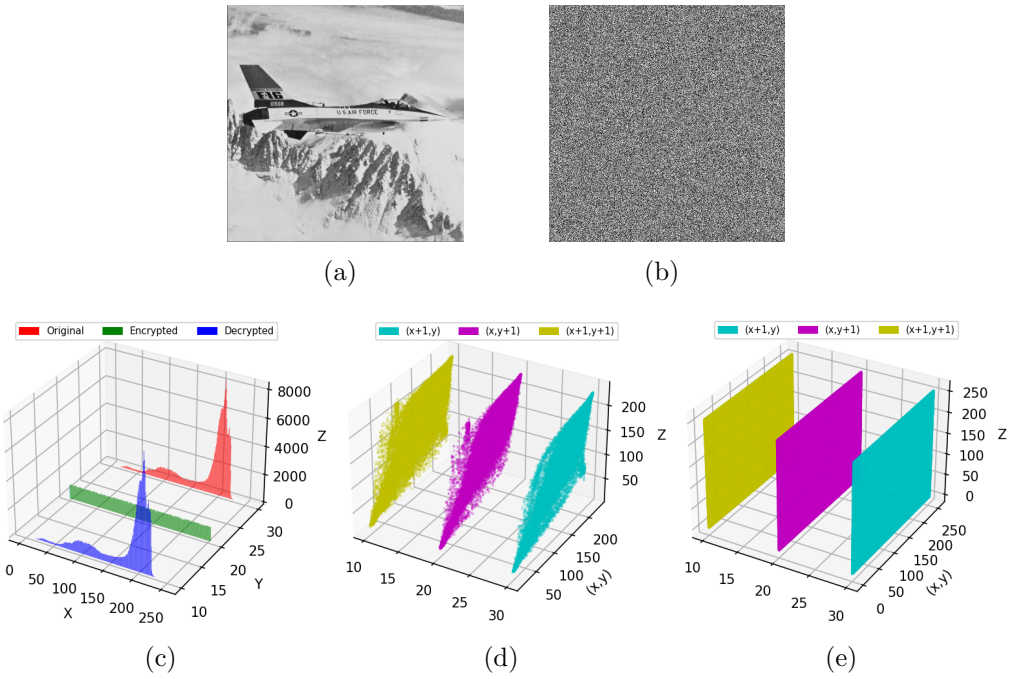


Figure 5.11: Airplane (a) Plain image (b) Encrypted image (c) Histogram (plain, encrypted and decrypted image) (d) Correlation between adjacent pixels (plain image) (e) Correlation between adjacent pixels (encrypted image)

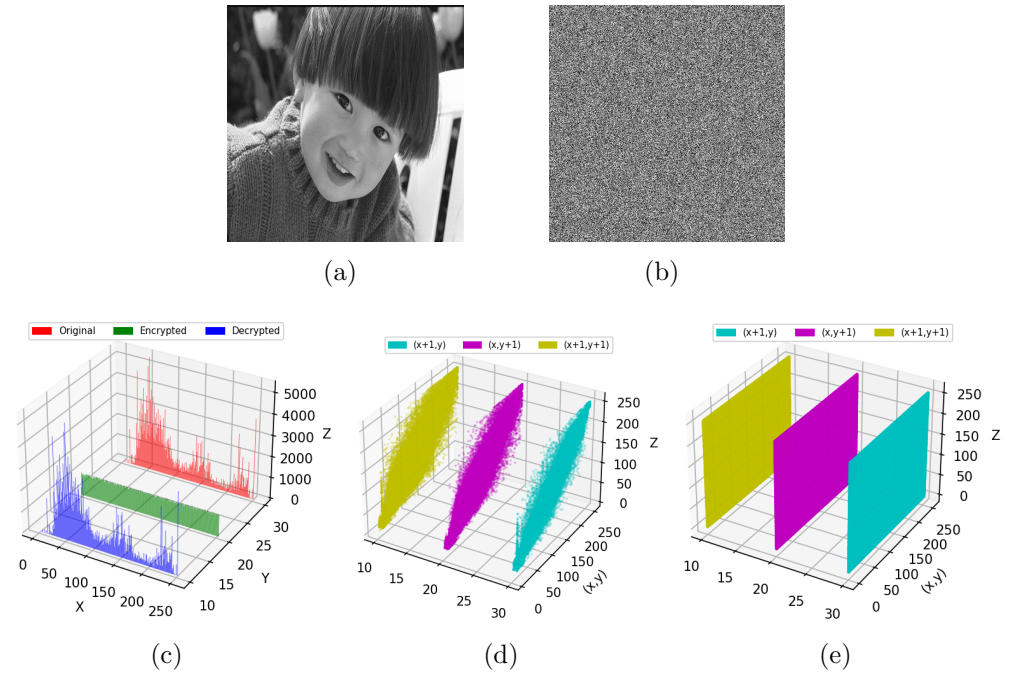


Figure 5.12: Boy (a) Plain image (b) Encrypted image (c) Histogram (plain, encrypted and decrypted image) (d) Correlation between adjacent pixels (plain image) (e) Correlation between adjacent pixels (encrypted image)

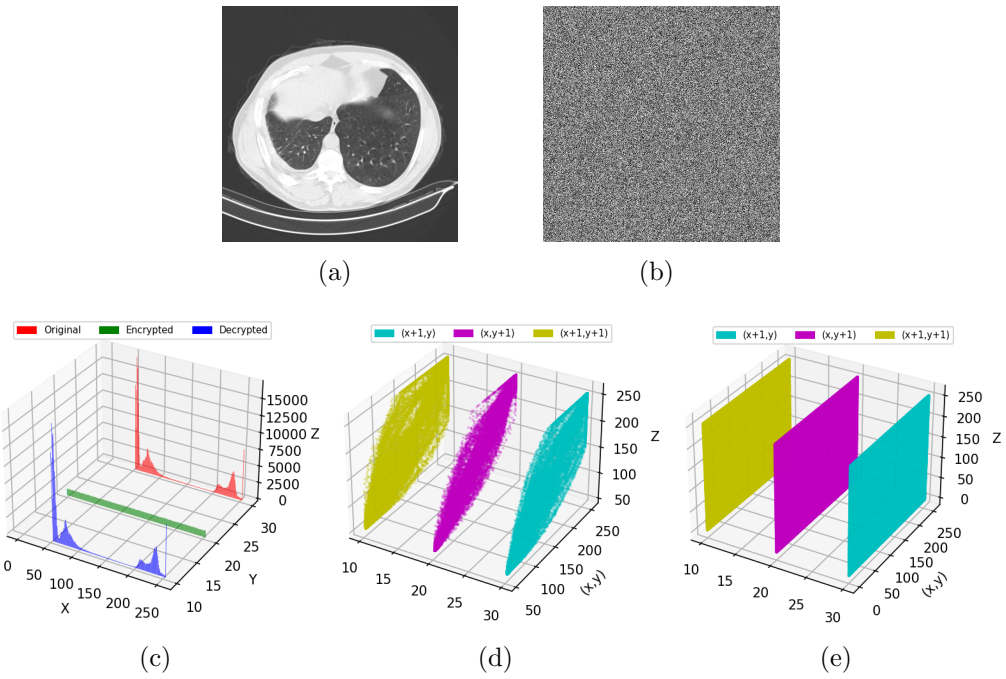


Figure 5.13: CT-Covid (a) Plain image (b) Encrypted image (c) Histogram (plain, encrypted and decrypted image) (d) Correlation between adjacent pixels (plain image) (e) Correlation between adjacent pixels (encrypted image)

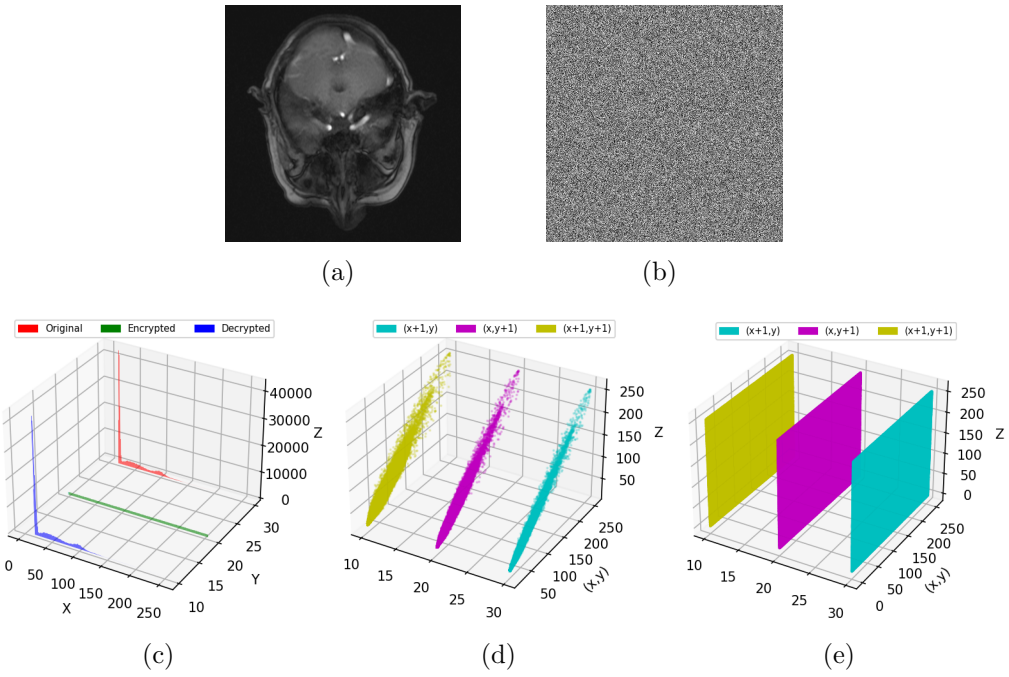
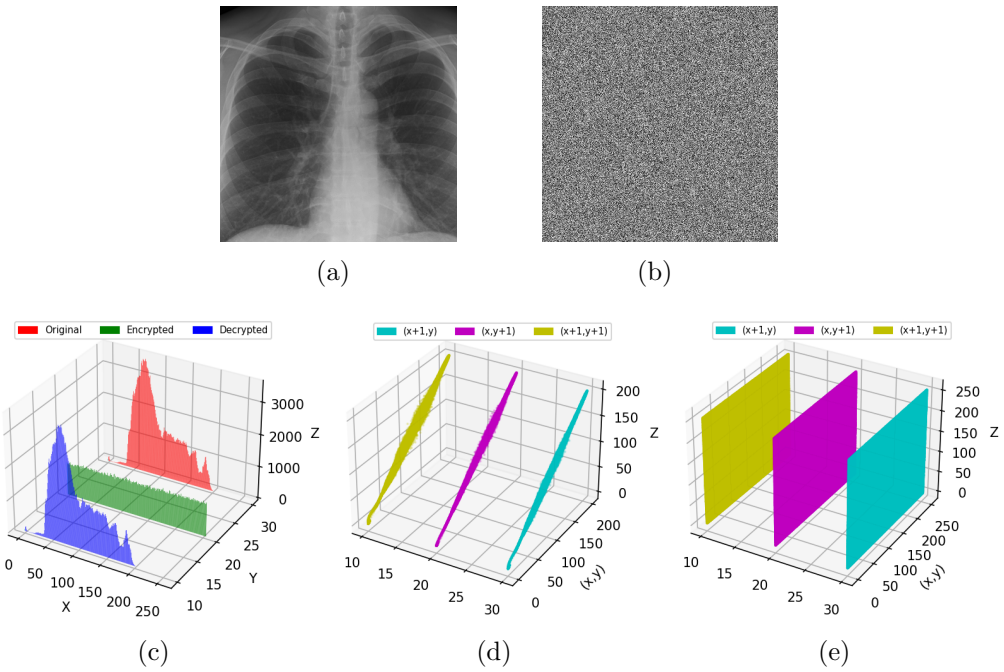


Figure 5.14: MRI-Brain (a) Plain image (b) Encrypted image (c) Histogram (plain, encrypted and decrypted image) (d) Correlation between adjacent pixels (plain image) (e) Correlation between adjacent pixels (encrypted image)



(a) (b) (c) (d) (e)

Figure 5.15: Xray-Chest (a) Plain image (b) Encrypted image (c) Histogram (plain, encrypted and decrypted image) (d) Correlation between adjacent pixels (plain image) (e)Correlation between adjacent pixels (encrypted image)

PSNR, MSE, and SSIM metrics are used to evaluate the overall process's performance [215]. SSIM metric is calculated as follows:

$$\mathrm{SSIM}(\mathbf{O},\mathbf{C}) = \frac{(2\mu_O\mu_C + \Psi_1)(2\sigma_{OC} + \Psi_2)}{(\mu_O^2 + \mu_C^2 + \Psi_1)\,(\sigma_O^2 + \sigma_C^2 + \Psi_2)} \tag{5.28}$$

where O_i and C_i are the i^{th} pixel in the original(img1) and transformed image (img2). Ψ_1 and Ψ_2 denotes regularize parameters. The other two metrics are computed using (5.19). The results of a qualitative and quantitative evaluation of image encryption and decryption using the proposed optimal-key-based AES technique are shown in Figures 5.11–5.15 and Table 5.6 ('↑' indicates higher value and '↓' presents lower value are preferable). The cumulative performance of the proposed MDO is examined on several images and also compared using two distinct fitness functions. MSE, $PSNR$, and $SSIM$ values are computed between the pair of (original, encrypted) images and (original, decrypted) images. The quantitative results of performance metrics between

the pair of original and encrypted images are shown in Table 5.6 while considering visual metric and gap test as fitness functions. By utilizing $PSNR$ as a fitness function, we can see that optimal key generation using the proposed MDO yields the highest MSE and least $PSNR$ between original and encrypted images. In addition, the encrypted image has the lowest $SSIM$ value when compared to the original image. The effect of considered objective functions for optimal key generation for image encryption shows that in both cases, we are capable of designing a lossless encryption scheme, but overall, MDO with visual metric as an objective function outperformed the MDO with gap test for the encryption scheme.

Furthermore, for a better encryption method, $PSNR$ and $SSIM$ should be as low as possible, and MSE between the original and encrypted image should be as high as possible. Similarly, with lossless decryption, the SSIM value is one, and the $PSNR$ value is infinite. The findings clearly show that the proposed MDO algorithm achieved high MSE, low $PSNR$, and low $SSIM$ for all original-encrypted images. Moreover, we obtained SSIM Value 1, $MSE = 0$, and $PSNR = infinity$ for all original-decrypted images. As a result, the optimal key selected provides the attacker with the least amount of information while keeping the quality of the decrypted images. This demonstrates the consistency of our proposed MDO algorithm.

5.8.3 Performance Analysis of Proposed MDO for Feature Selection

Following the establishment of the MDO against well-known test functions and a successful validation against nine other pre-established optimization algorithms, it was examined on the problem of feature selection on 19 different datasets taken from UCI repository [1], the details of which are tabulated in Table 5.7. The datasets are of varying sizes in terms of the number of instances, the number of features, and the number of classes, as seen from Table 5.7. Five different classifiers, namely, KNN, RBF-SVM, De-

[1] https://archive.ics.uci.edu/ml/index.php

Table 5.7: Dataset description

Dataset No.	Dataset Name	#Features	#Instances	#Classes
D1	Breast Cancer Wisconsin (Diagnostic)	30	569	2
D2	Cervical Cancer (Risk Factors)	35	858	2
D3	Diabetes	8	768	2
D4	Glass	9	214	7
D5	Heart-Statlog	13	270	2
D6	Ionosphere	34	351	2
D7	Sonar	60	208	2
D8	Thyroid	21	7200	3
D9	Wine	13	178	3
D10	Chess (King-Rook vs. King-Pawn)	36	3196	2
D11	Statlog (Australian Credit Approval)	14	690	2
D12	Zoo	16	101	7
D13	Parkinson	23	195	2
D14	Cardiotocography	23	2126	3
D15	Hepatitis	19	155	2
D16	Lung Cancer	56	32	3
D17	Thoracic Surgery	17	470	2
D18	QSAR biodegradation	41	1055	2

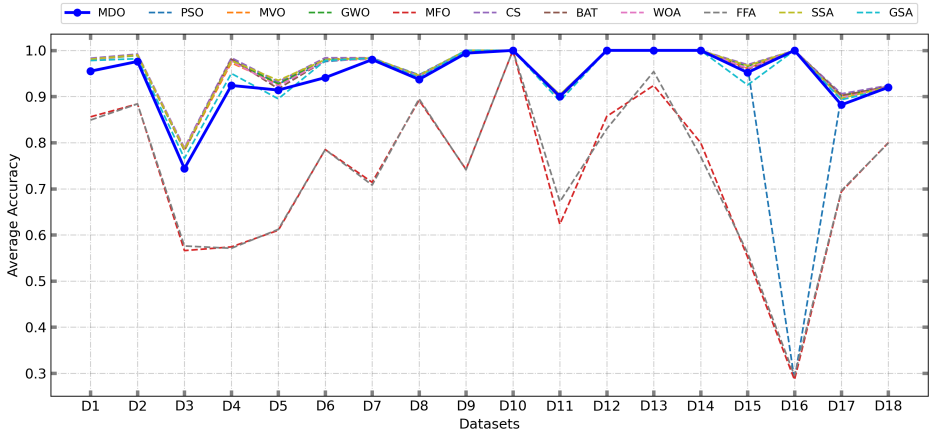
cision Tree (DT), Multi-layer Perceptron (MLP), and Extra Tree (ET), were used as a learning method. The population size was initialized to 40, and the termination criteria were set to (i) reaching a pre-defined maximum accuracy of 100% or (ii) reaching the maximum number of iterations, which was set to 50 in this case. The algorithm was run for a total of 30 times for each dataset, with each classifier, and the average values were reported.

For the purpose of validating the MDO with respect to the problem of feature selection, a comparison was made with ten pre-established optimizers. These optimizers were applied to the same 18 datasets with the same five classifiers. The pre-established optimizers used for comparison are as follows: PSO, MVO, GWO, MFO, CS, BAT,

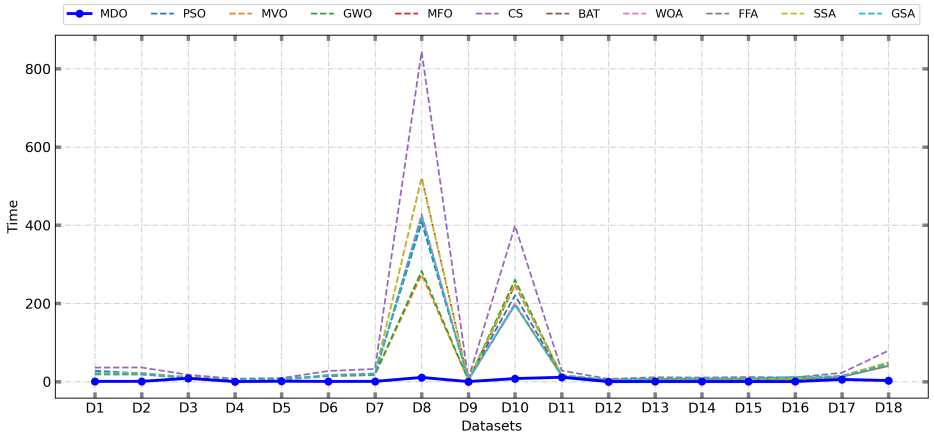
WOA, FFA, SSA, and GSA. For proper comparison, the population size for all algorithms is fixed at 40, the maximum number of iterations was kept the same as in the previous case, and the number of times the whole experiment runs was kept at 30. An in-depth analysis has been performed to check the efficacy of MDO for feature selections. The graphical analysis for 18 datasets with ten optimizers using five different classifiers has been shown in Figure 5.16 to Figure 5.20. The average accuracy, number of selected features, and execution time using the KNN classifier have been shown in Figure 5.16.

Similarly, obtained values in terms of accuracy, the number of features, and execution time with RBF-SVM are presented in Figure 5.17. Further, Figure 5.18 to Figure 5.20 show a similar presentation with MLP, ET, and DT, respectively. It is visible that MDO has achieved effectiveness in computation time and gives superior performance compared to other state-of-the-art methods. However, on a few datasets, it achieves superiority in terms of time and features by sacrificing minor accuracy, which may vary from 1 to 2%. With MLP, MDO achieves 98% accuracy in 22 sec, while the highest accuracy, 99.7%, is achieved by WOA in 583.40 sec. Similarly, on D5, MDO gives 94% accuracy with average 5.4 features in 26.89 sec, while the highest accuracy is 94.93% with average 5.6 features achieved by CS in 492 sec. MDO attained 99.05% accuracy with average 10.5 features in 139.45 sec, while the highest accuracy attained on the same dataset D8 by CS was 99.14% with 11.43 average features in 7171.98 sec.

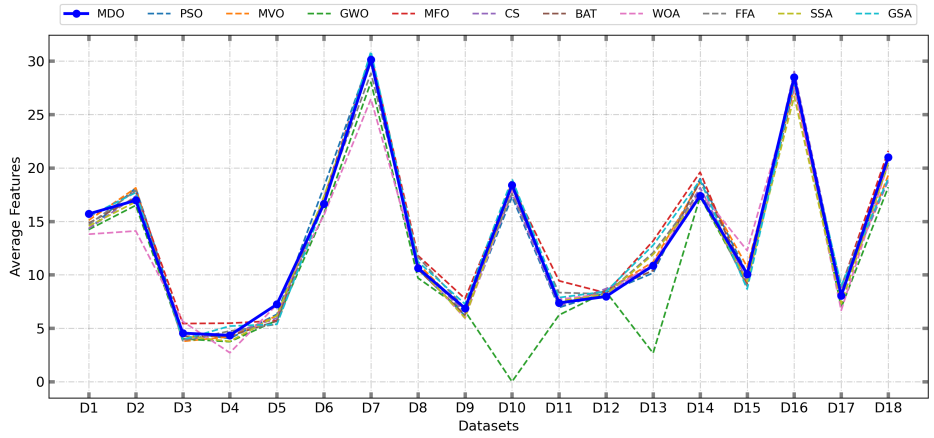
On the other hand, MDO, CS, and GWO all achieved accuracy 100% on D9 with average features 6.2, 6.5, and 6 features in 64 sec, 324.11 sec, and 162.21 sec respectively. Further on D12, MDO reached 99.80% accuracy with 7.9 average features in 2.95 sec. On the same data with MLP, except for MFO and FFA, all others achieved 100% accuracy with features (8.2 - 9.6) in time ranging from 75 - 152 sec. MDO got 100% on D13 with minimum features 11.13 in 0.762 sec while others reached with (10-12) features in (18-108.25) sec. In another case (D14), MDO reached 99.8% accuracy with



(a)

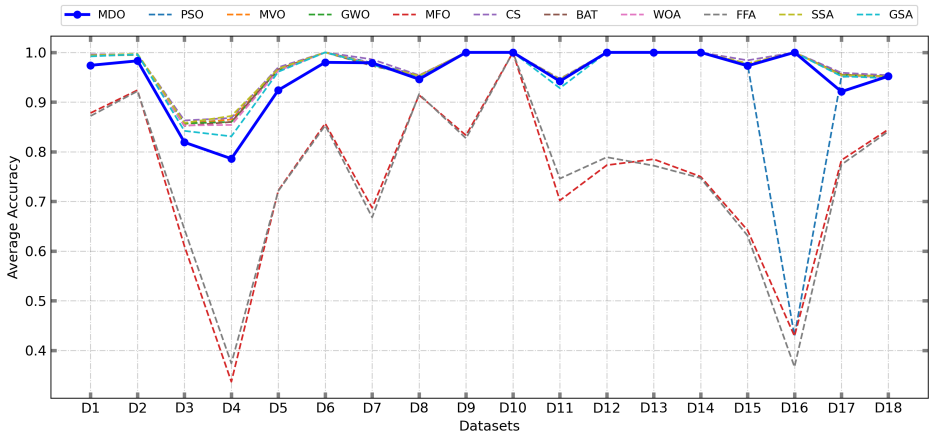


(b)

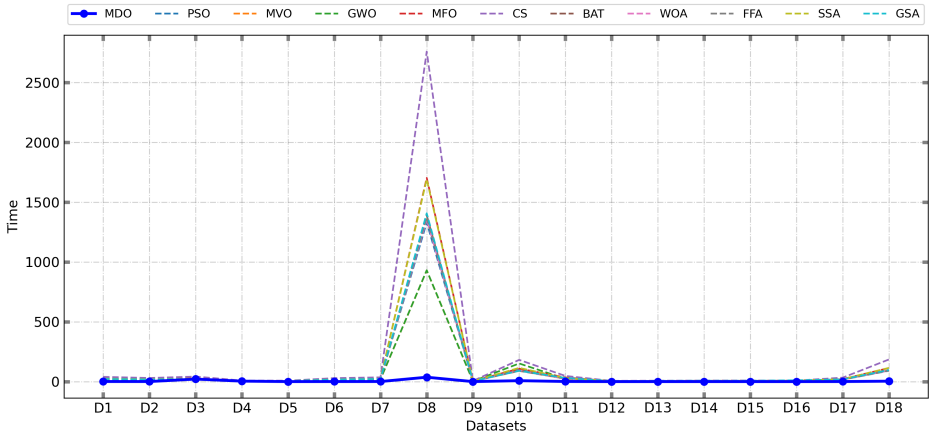


(c)

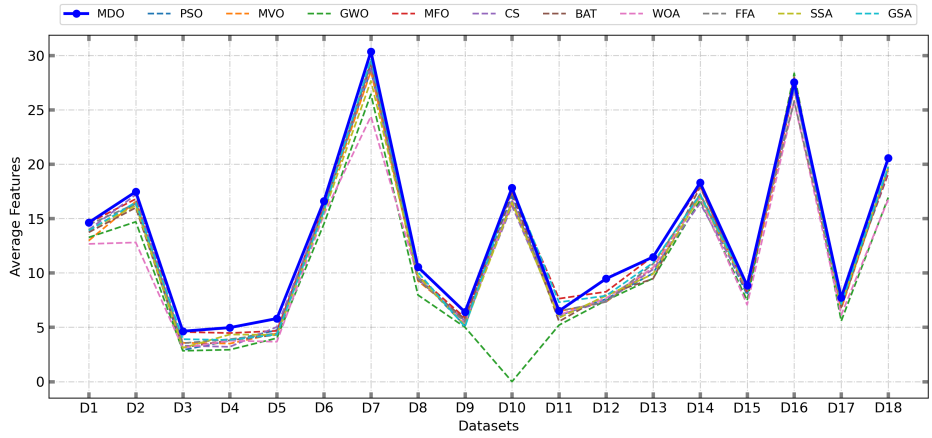
Figure 5.16: Line plots for comparative study of average test accuracy, computation time, and number of selected features on 18 datasets for MDO and other optimizers with KNN classifier



(a)

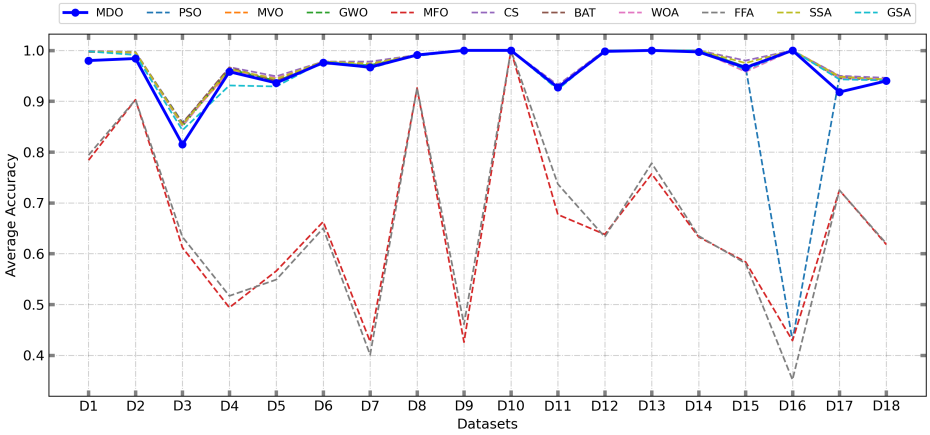


(b)

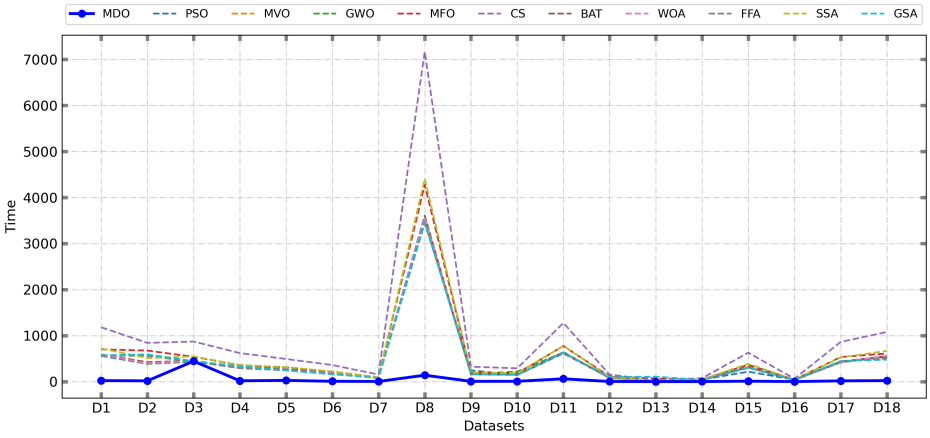


(c)

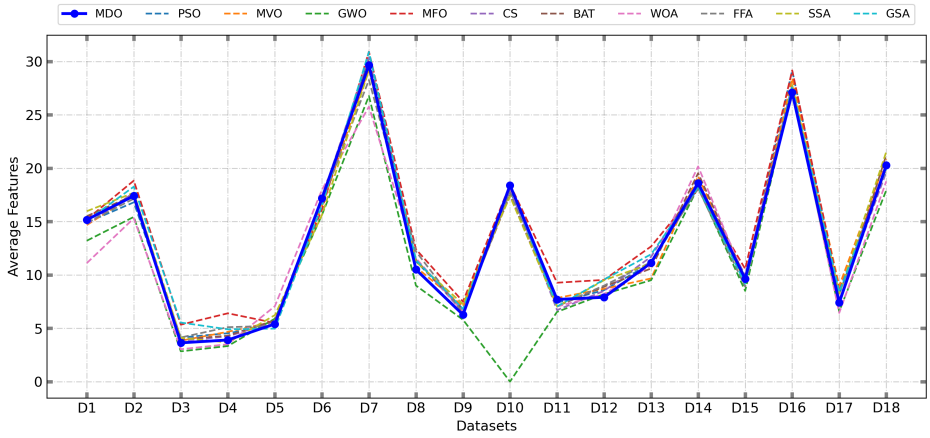
Figure 5.17: Line plots for comparative study of average test accuracy, computation time, and number of selected features on 18 datasets for MDO and other optimizers with SVM (RBF) classifier



(a)

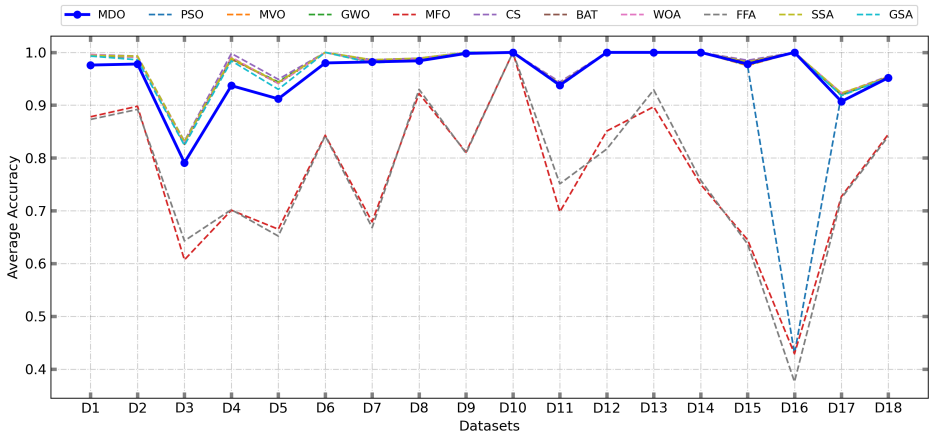


(b)

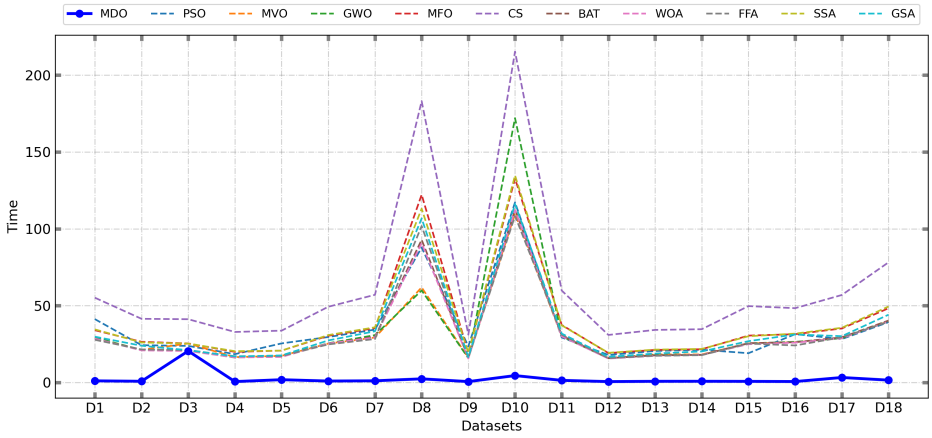


(c)

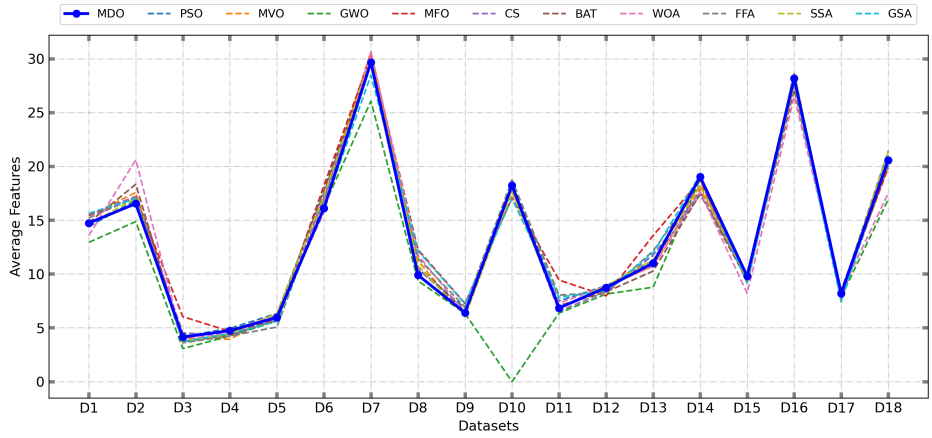
Figure 5.18: Line plots for comparative study of average test accuracy, computation time, and number of selected features on 18 datasets for MDO and other optimizers with MLP classifier



(a)

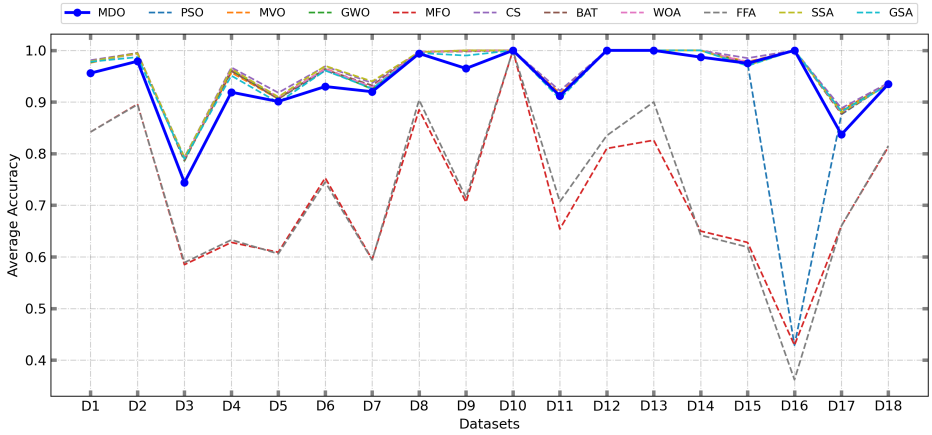


(b)

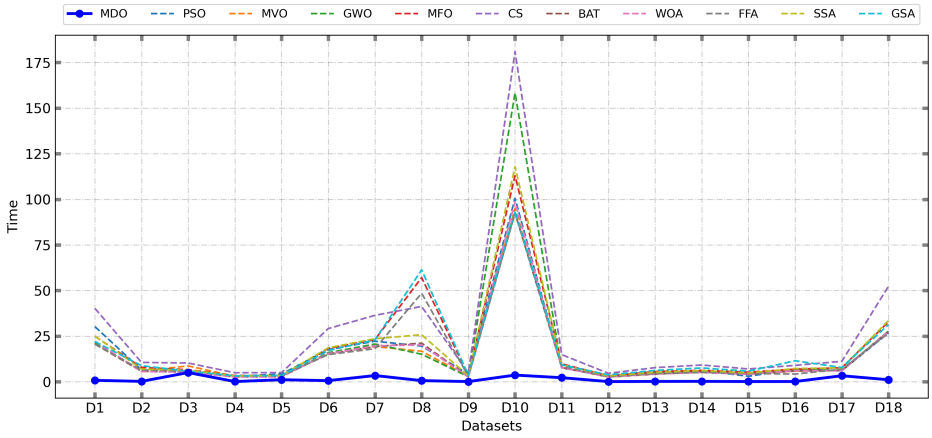


(c)

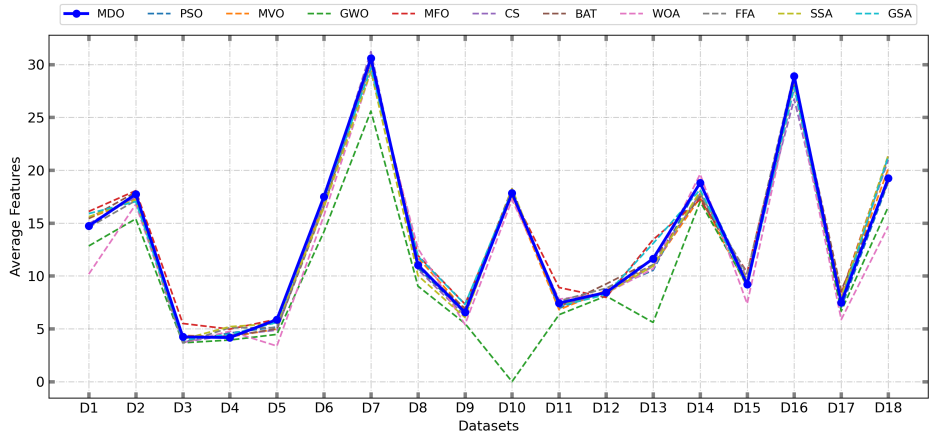
Figure 5.19: Line plots for comparative study of average test accuracy, computation time, and number of selected features on 18 datasets for MDO and other optimizers with Extra Tree classifier



(a)



(b)



(c)

Figure 5.20: Line plots for comparative study of average test accuracy, computation time, and number of selected features on 18 datasets for MDO and other optimizers with Decision Tree classifier

18.6 average features in a very short time of 1.54 sec while SSA, CS, and BAT give 100% performance with 18.46, 18.7, 19.5 average number of features in 48.7, 75.01, and 40.43 sec respectively. Other trends have been shown with different classifiers in Figure 5.16 to Figure 5.20.

Table 5.8 also includes a win-tie-lose analysis that takes accuracy and time into account. Each entry in the tables has a triplet value. The first value of the triplet indicates how many times the method in the row dominates the method defined in the column.

Table 5.8: Win-Tie-Lose (Accuracy-Time)

	PSO	MVO	GWO	MFO	CS	BAT	WOA	FFA	SSA	GSA
MDO_MLP	[7, 10, 1]	[8, 9, 1]	[7, 10, 1]	[18, 0, 0]	[5, 13, 0]	[7, 11, 0]	[8, 10, 0]	[17, 1, 0]	[6, 12, 0]	[9, 9, 0]
MDO_SVM	[8, 10, 0]	[8, 10, 0]	[9, 8, 1]	[18, 0, 0]	[6, 12, 0]	[9, 9, 0]	[9, 9, 0]	[18, 0, 0]	[8, 10, 0]	[10, 8, 0]
MDO_ET	[6, 12, 0]	[8, 10, 0]	[7, 11, 0]	[18, 0, 0]	[5, 13, 0]	[6, 12, 0]	[7, 10, 1]	[18, 0, 0]	[5, 13, 0]	[9, 9, 0]
MDO_DT	[5, 13, 0]	[5, 13, 0]	[7, 11, 0]	[18, 0, 0]	[4, 14, 0]	[6, 12, 0]	[6, 12, 0]	[18, 0, 0]	[6, 12, 0]	[8, 10, 0]
MDO_KNN	[6, 12, 0]	[7, 11, 0]	[7, 11, 0]	[18, 0, 0]	[5, 13, 0]	[6, 12, 0]	[5, 13, 0]	[18, 0, 0]	[5, 13, 0]	[9, 9, 0]

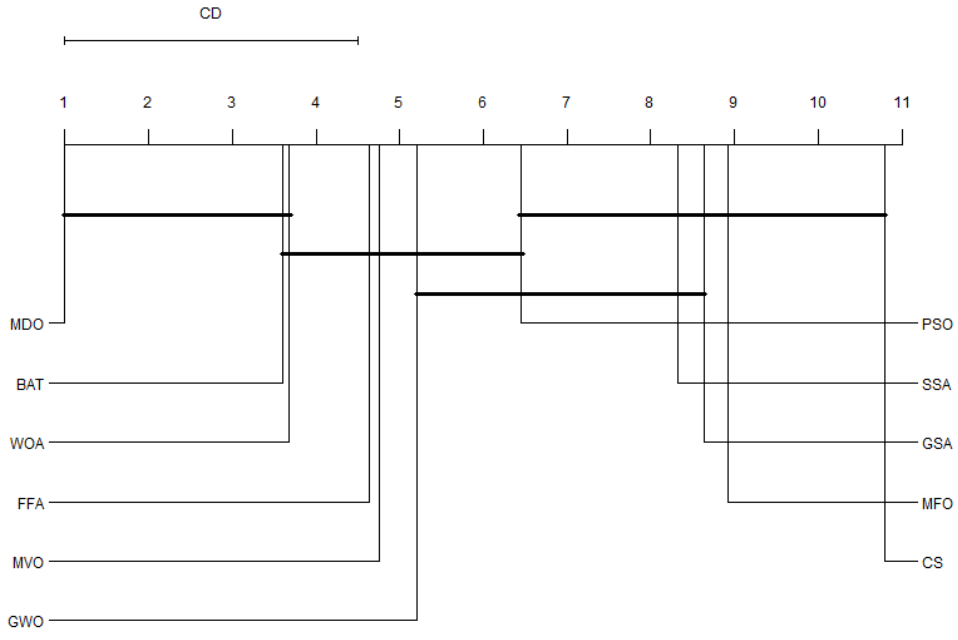


Figure 5.21: Critical difference diagram for average computation time. The goodness of the model is ranked from best to worst (left to right), and the bold lines indicate methods that do not produce statistically significant differences.

The second value indicates the tie condition or no significant difference. The third value indicates how many times the method defined in the row is dominated by the method defined in the column. Table 5.8 shows that, with a few exceptions, MDO outperforms or is competitive with other optimizers in the majority of cases. The comparison of these methods is further analyzed using a critical difference diagram, as shown in Figure 5.21 for the Decision Tree classifier. This graph shows the average ranking as well as pairwise differences between optimizers by considering average computation time. The Nemenyi post-hoc statistical test [256] is performed with a significance level of 0.05 to draw the critical difference diagram. The optimizer with the lowest computation time is given a higher rank, and the same is shown in Figure 5.21. Results from all datasets demonstrate that, in terms of computation time, MDO outperforms all optimizers with all five classifiers. It shows its suitability for low-cost devices and is able to solve the issue of being computationally expensive due to its efficient search mechanism and good initialization. It has few tunable parameters and is simple to understand, thus useful for researchers with a non-EC background.

5.8.4 Experimental Verification of the Effectiveness of MDO-M

5.8.4.1 Applications to Benchmark Test Problems

The proposed algorithm MDO-M was tested against ZDT test problems, Fonseca-Fleming function (FOF), Schaffer functions(SCH), and Poloni's function(POL) involving two objectives each for minimization (both) [267] which are described as in chapter 3.

(i) Zitzler-Deb-Thiele's function N.1

$$ZDT1 = Minimize \begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)h(f_1(x), g(x)) \\ g(x) = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i, \, where \, x_i \in [0,1], i \in [1,30] \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} \end{cases} \tag{5.29}$$

(ii) Zitzler-Deb-Thiele's function N.2

$$ZDT2 = Minimize \begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)h(f_1(x), g(x)) \\ g(x) = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i, \quad where\, x_i \in [0,1], i \in [1,30] \\ h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2 \end{cases} \tag{5.30}$$

(iii) Zitzler-Deb-Thiele's function N.3

$$ZDT3 = Minimize \begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)h(f_1(x), g(x)) \\ g(x) = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i, \, where\, x_i \in [0,1], i \in [1,30] \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \frac{f_1(x)}{g(x)} sin(10\pi f_1(x)) \end{cases} \tag{5.31}$$

(iv) Zitzler-Deb-Thiele's function N.4

$$ZDT4 = Minimize \begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)h(f_1(x), g(x)) \\ g(x) = 91 + \sum_{i=2}^{10} (x_i^2 - 10cos(4\pi x_i)), \, where\, x_1 \in [0,1], x_i \in [-5,5], i \in [2,10] \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} \end{cases} \tag{5.32}$$

(v) Zitzler-Deb-Thiele's function N.6

$$ZDT6 = Minimize \begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)h(f_1(x), g(x)) \\ g(x) = 91 + \sum_{i=2}^{10} (x_i^2 - 10cos(4\pi x_i)), \, where\, x_1 \in [0,1], x_i \in [-5,5], i \in [2,10] \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} \end{cases} \tag{5.33}$$

(vi) Fonseca-Fleming function

$$Minimize \begin{cases} f_1(x) = 1 - exp\left[-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right] \\ \\ f_2(x) = 1 - exp\left[-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right], where\, x_i \in [-4,4], i \in [1,n] \end{cases} \tag{5.34}$$

(vii) Schaffer function N.1

$$Minimize \begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2, \; where \, x \in [-10, 10] \end{cases} \tag{5.35}$$

(viii) Schaffer function N.2

$$\alpha(x) = \begin{cases} f_1(x) = \begin{cases} -x, & if \, x \leq 1 \\ x-2, & if \, 1 < x \leq 3 \\ 4-x, & if \, 3 < x \leq 4 \quad where \quad x \in [-5, 10] \\ x-4, & if \, x > 4 \end{cases} \\ f_2(x) = (x-5)^2 \end{cases} \tag{5.36}$$

(ix) Poloni's two objective function

$$\alpha(x) = \begin{cases} f_1(x,y) = [1 + (A_1 - B_1(x,y))^2 + (A_2 - B_2(x,y))^2] \\ f_2(x,y) = (x+3)^2 + (y+1)^2, \quad where \; x, y \in [-\pi, \pi] \end{cases}$$

$$where = \begin{cases} A_1 = 0.5sin(1) - 2cos(1) + sin(2) - 1.5cos(2) \\ A_2 = 1.5sin(1) - cos(1) + 2sin(2) - 0.5cos(2) \\ B_1(x,y) = 0.5sin(x) - 2cos(x) + sin(y) - 1.5cos(y) \\ B_2(x,y) = 1.5sin(x) - cos(x) + 2sin(y) - 0.5cos(y) \end{cases} \tag{5.37}$$

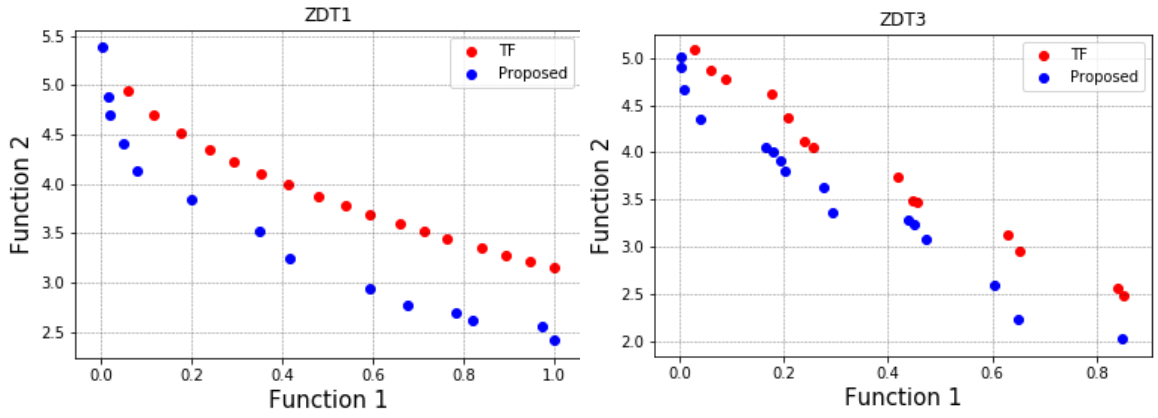


Figure 5.22: Comparison of True Pareto Front and the Pareto Front obtained by proposed MDO-M for (a) ZDT1 (Left) and (b) ZDT3 (Right)

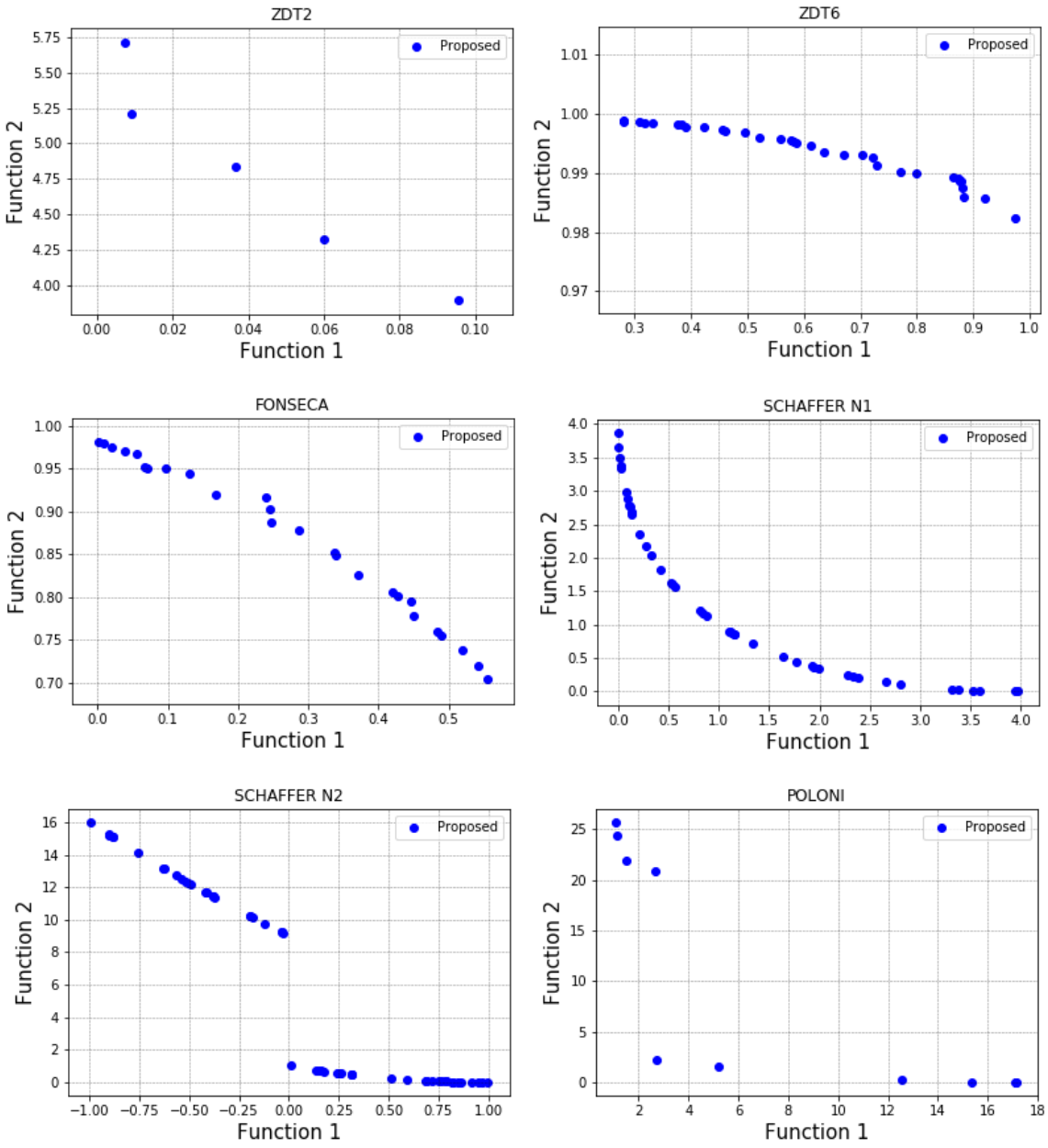


Figure 5.23: Pareto Fronts obtained by proposed MDO-M for ZDT2, ZDT6, FONSECA, SCHAFFER N1, SCHAFFER N2 and POLONI

The MDO-M was initialized with a population size of 400, and the maximum number of iterations was set to 100. The algorithm was run 10 times for each test problem, and the results were averaged. The Pareto fronts obtained by the proposed algorithm against each of the nine test problems and some of them are shown in Figure 5.22 and

Figure 5.23. Figure 5.22 (a) and 5.22 (b) show a side-by-side plot of the True Front and the Pareto front obtained by the proposed algorithm for test problems ZDT1 and ZDT3, respectively. It can be seen that the obtained Pareto fronts approximate the actual true Pareto front with significant precision. Figure 5.23 presents the Pareto fronts obtained by the proposed algorithm MDO-M for the rest of the test suits. It must be noted that the MDO-M returns significantly accurate Pareto fronts for test problems ZDT1, ZDT3, ZDT6, FOF, SCH, and POL. Further, it is also tested on DTLZ test suits [268].

A number of different metrics [269] such as Spacing (S), Maximum Pareto Front Error (ME), Inverted Generational Distance (IGD), and Hypervolume(HV) were used to measure the performance of the MDO-M on the benchmark test problems. To validate the same, the results were compared with those of existing multi-objective

Table 5.9: Spacing performance metric

Test	**Proposed**		**NSGA-II**		**OMOPSO**	
Function	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
ZDT1	**2.880e-17**	3.22e-17	5.356e-16	2.19e-16	3.831e-17	5.42e-17
ZDT2	**0.0**	0.0	3.235e-16	2.04e-16	1.911e-17	2.70e-17
ZDT3	**0.0**	0.0	7.066e-16	3.44e-16	3.976e-17	2.81e-17
ZDT4	**5.822e-17**	6.11e-17	3.905e-16	5.52e-16	9.593e-17	0.013
ZDT6	5.639e-17	2.23e-17	6.137e-16	2.27e-16	**0.0**	0.0

Table 5.10: Maximum Pareto Front Error performance metric

Test	**Proposed**		**NSGA-II**		**OMOPSO**	
Function	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
ZDT1	**6.3042**	0.014	14.7710	0.022	14.6197	0.015
ZDT2	**3.6648**	0.024	12.3821	0.511	13.2519	0.014
ZDT3	**8.8412**	0.007	20.2436	0.728	20.1595	0.226
ZDT4	29.3711	0.392	**5.23419**	0.476	6.72347	0.647
ZDT6	41.2449	0.105	**39.4841**	0.642	44.7268	0.381

Table 5.11: Inverted Generational Distance performance metric

Test	Proposed		NSGA-II		OMOPSO	
Function	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
ZDT1	**0.00264**	0.0002	0.01846	0.0001	0.12290	0.0011
ZDT2	0.02445	0.0073	**0.01205**	0.0001	0.07821	0.0024
ZDT3	**0.00342**	0.0293	0.02039	0.0005	0.24570	0.0310
ZDT4	0.01554	0.1002	**0.00562**	0.0079	0.12491	0.0134
ZDT6	**0.00013**	0.0024	0.02645	2.54 e-8	0.52597	0.0044
DTLZ1	0.021	0.015	**0.018**	0.021	0.035	0.014
DTLZ2	**0.013**	0.007	0.022	0.011	0.031	0.019
DTLZ3	0.025	0.007	0.024	0.017	**0.021**	0.006
DTLZ4	0.024	0.012	**0.021**	0.013	0.036	0.034

Table 5.12: Hypervolume performance metric

Test	Proposed		NSGA-II		OMOPSO	
Function	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
ZDT1	**5.4299**	**2.32**	5.1018	2.11	4.9375	2.26
ZDT2	0.0954	0.062	**1.6993**	**1.09**	1.3047	0.81
ZDT3	**2.40**	**0.41**	2.1268	0.69	1.9674	0.27
ZDT4	53.128	4.76	**56.004**	**3.86**	54.081	2.08
ZDT6	**0.1049**	**0.048**	0.0825	0.051	0.0667	0.092
DTLZ1	14.61e4	3.08e4	**21.41e4**	**2.28e4**	9.263e4	1.10e4
DTLZ2	**21.780**	**2.62**	13.974	2.36	14.433	2.18
DTLZ3	17.03e4	2.91e4	**22.39e4**	**2.23e4**	14.59e4	1.01e4
DTLZ4	**16.681**	**2.43**	11.007	1.07	4.145	2.21

optimizers like NSGA-II [270], and OMOPSO [271]. Ten sets of results were obtained for each optimizer, and the averaged values of the performance metrics were reported along with the standard deviation values as described in Tables 5.9 to Table 5.12. *IGD* is a variation of Generational Distance (GD) where *GD* reports how far, on average,

the obtained Pareto front is from the true Pareto front and is defined in (5.38).

$$GD \cong \frac{(\sum_{i=1}^{n} d_i^p)^{\frac{1}{p}}}{|PF_{obtained}|} \tag{5.38}$$

where, $|PF_{obtained}|$ denotes the number of vectors in $PF_{obtained}$, $p = 2$ and di is the Euclidean phenotypic distance between each member, *'i'*, of $PF_{obtained}$ and the closest member in PF_{true} to that member *'i'*. When $GD = 0$ we have $PF_{obtained} = PF_{true}$. IGD requires the true Pareto front to be known beforehand. It is known to be a convergence metric. The spacing (S) metric is used to measure diversity and numerically describes the spread of the vectors in $PF_{obtained}$. It is defined in (5.39).

$$S \cong \sqrt{\frac{1}{|PF_{obtained}| - 1} \sum_{i=1}^{|PF_{obtained}|} (\overline{d} - d_i)^2} \tag{5.39}$$

$$where, d_i = min_j \left(\left| f_1^i(x) - f_1^j(x) \right| + \left| f_2^i(x) - f_2^j(x) \right| \right), i, j = 1..n$$

and, $\overline{d}$ is the mean of all d_i while n is the number of vectors in $PF_{obtained}$. When $S = 0$, all members are spaced evenly apart. Further, ME measures the largest minimum distance between each vector in $PF_{obtained}$ and the corresponding closest vector in PF_{true}. It is described in (5.40).

$$ME \cong max_j \left\{ min_i \left(\sum_{k=1}^{m} \left| f_k^i(x) - f_k^j(x)) \right|^p \right)^{\frac{1}{p}} \right\} \tag{5.40}$$

Where, $i = 1, .., |PF_{obtained1}|$ and $j = 1, .., |PF_{obtained2}|$ index vectors in $PF_{obtained}$ and PF_{true} respectively. A value of '0' for ME indicates $PF_{obtained} \subseteq PF_{true}$ whereas any other value indicates at least one vector in $PF_{obtained}$ is not in PF_{true}.

It can be seen from Table 5.9 to Table 5.12 that in terms of IGD performance metric, the MDO-M completely outperforms OMOPSO except on DTLZ3, whereas it outperforms NSGA-II on all except four (ZDT2, ZDT4, DTLZ1, DTLZ4) cases. In

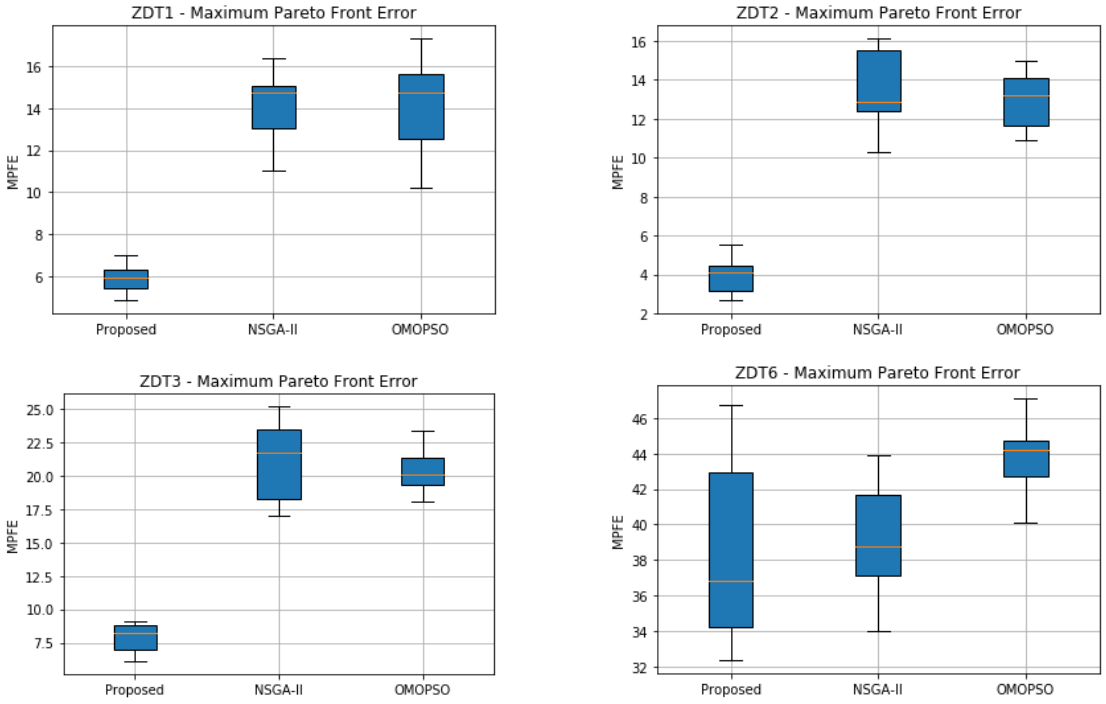


Figure 5.24: Box plots for maximum Pareto front error values obtained by proposed MDO-M, NSGA-II, and OMOPSO on ZDT1, ZDT2, ZDT3, and ZDT6.

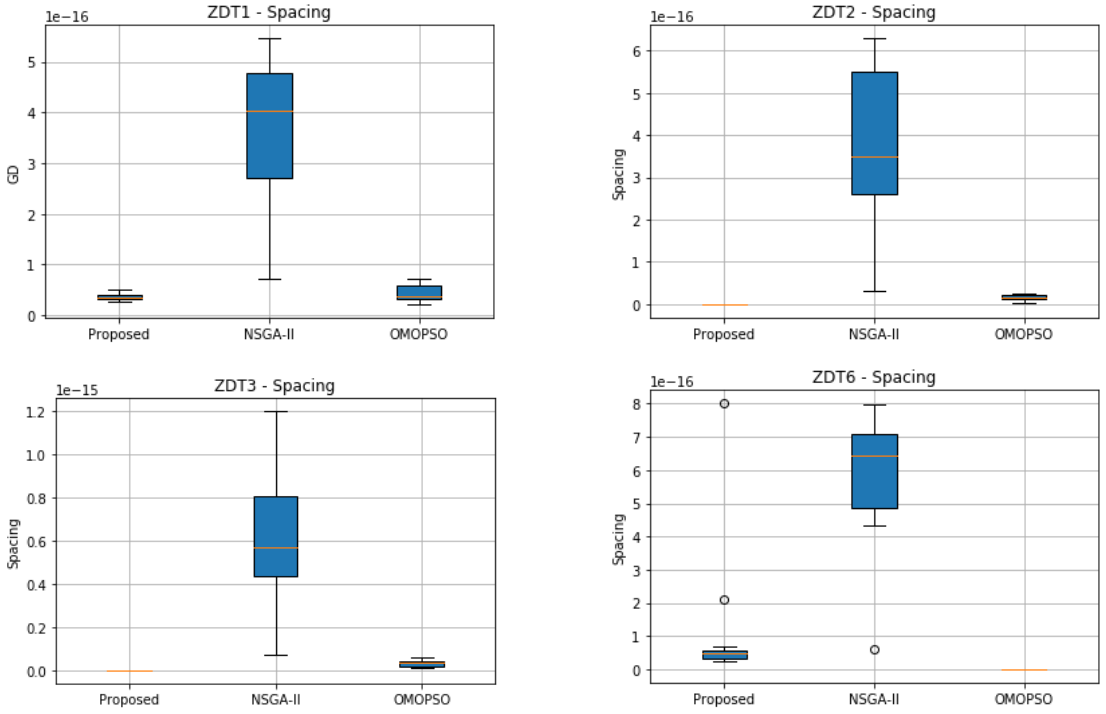


Figure 5.25: Box plots for spacing (S) values obtained by proposed MDO-M, NSGA-II, and OMOPSO on ZDT1, ZDT2, ZDT3, and ZDT6.

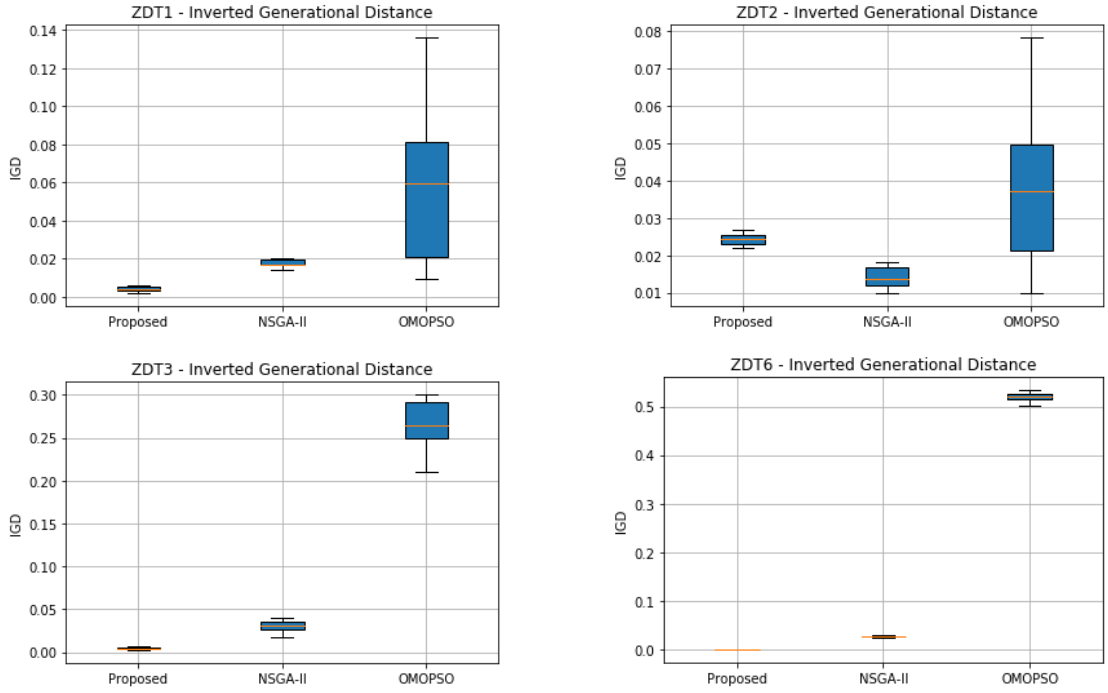


Figure 5.26: Box plots for IGD values obtained by proposed MDO-M, NSGA-II, and OMOPSO on ZDT1, ZDT2, ZDT3, and ZDT6.

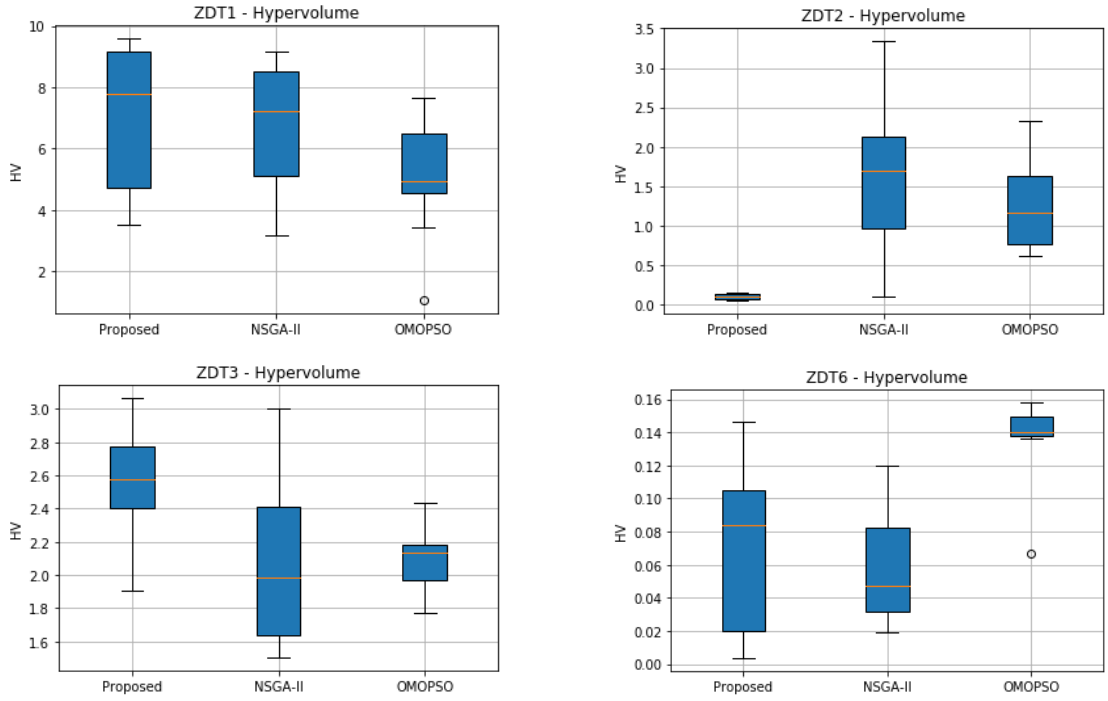


Figure 5.27: Box plots for hypervolume (HV) values obtained by proposed MDO-M, NSGA-II, and OMOPSO on ZDT1, ZDT2, ZDT3, and ZDT6.

the case of the S performance metric, MDO-M is evidently significantly better than both NSGA-II and OMOPSO. However, in terms of the ME performance metric, it outperforms OMOPSO and is better than NSGA-II in all except two cases, and finally, in terms of HV, MDO-M completely outperforms OMOPSO and performs better than NSGA-II in five cases. Boxplots of some test functions are shown in Figure 5.24, Figure 5.25, and Figure 5.26 for the obtained performance metrics ME, S, and IGD, respectively. In addition, a boxplot for HV was shown in Figure 5.27. Overall, it can be said that MDO-M is clearly better than OMOPSO and sufficiently comparable to NSGA-II.

5.8.4.2 Application to Feature Selections and Classification

To provide future insights into the MDO-M performance on feature selection, the proposed MDO-M was applied to the problem of feature selections on the subset of datasets (9 datasets) used in the case of our MDO algorithm. Two classifiers, namely SVM (RBF) and MLP classifier, were used. The two objectives for optimization were chosen to be (i) the error rate obtained after classification performed on the selected feature subset and (ii) the number of features selected, both of which are minimization problems and conflicting in nature. The algorithm was run ten times for each dataset and classifier, and the results were averaged. Table 5.13 presents the obtained results in terms of the maximum, minimum, and average error rate along with a number of selected features for SVM (RBF) and MLP, respectively.

It can be observed from Table 5.13 that the proposed MDO-M algorithm performs exceptionally well on the problem of feature selection. The best performance is obtained with the MLP classifier that displays an error rate in the range of 0.0 to 0.05 (or, in other words, an accuracy range of 95% to 100%) on 90% of the datasets taken into consideration. Moreover, the number of features selected is significantly lesser compared to that obtained by the single-objective counterpart of our algorithm. Besides, the

Table 5.13: Results for feature selection using MDO-M with different classifiers

Dataset	Classifier	Error			Features			
		Min	Avg	Max	Min	Avg	Max	Out of
Statlog Heart	SVM (RBF)	0.056	0.059	0.074	4	5.2	7	13
	MLP	0.052	0.054	0.056	2	4.33	8	13
Wine	SVM (RBF)	0.000	0.000	0.000	2	3.4	6	13
	MLP	0.0	0.034	0.056	3	4.25	6	13
BC Wisconsin (Diagnostic)	SVM (RBF)	0.026	0.026	0.026	7	15.0	21	30
	MLP	0.008	0.013	0.017	10	11.0	12	30
Pima Indians Diabetes	SVM (RBF)	0.142	0.154	0.162	1	4.2	6	8
	MLP	0.168	0.201	0.246	1	3.5	6	8
Sonar	SVM (RBF)	0.071	0.090	0.119	27	29.8	32	60
	MLP	0.014	0.076	0.142	20	22.0	25	60
Glass	SVM (RBF)	0.209	0.251	0.279	4	5.0	6	9
	MLP	0.046	0.126	0.186	2	3.85	6	9
Thyroid	SVM (RBF)	0.048	0.054	0.065	5	8.6	13	21
	MLP	0.011	0.019	0.030	13	7.44	4	21
Cervical Cancer (Risk Factors)	SVM (RBF)	0.005	0.005	0.005	16	18.8	23	35
	MLP	0.011	0.036	0.081	11	14.37	19	35
Ionosphere	SVM (RBF)	0.028	0.045	0.057	13	17.0	21	34
	MLP	0.042	0.085	0.142	10	13.5	19	34

results obtained with SVM (RBF) classifier show that the algorithm successfully attains an accuracy in the range of 98-100% five times each. It is thereby proven that our proposed algorithm displays successful optimization of both the objectives chosen for the concerned problem.

5.9 Summary

In this work, we proposed a novel EC algorithm based on the combined concepts of the natural phenomena viz., starling murmuration, V-shaped flight of migrating birds,

and the dispersive migration followed by certain species of birds. The algorithm was tested against 15 test functions, both unimodal and multimodal, and the results were compared with nine other pre-existing optimization algorithms. It was observed that the proposed algorithm attains significantly accurate results and is sufficiently efficient in terms of the time taken for execution as well as search space exploration when compared to the other algorithms. The performance of the proposed algorithm was also found to be either better than or on par with nine other optimizers with which it was compared. After validating the proposed MDO against benchmarks, we used it for medical data security using an innovative optimum cryptographic key generation approach in the first part of the chapter. Two different objective functions have been considered for optimal key generation using MDO, and their effect has been shown on the image encryption problem.

Later, the proposed algorithm was used to select features from 18 datasets with varying instance and attribute sizes. The results show that when used with the MLP classifier, the proposed algorithm achieves good accuracy on all 18 datasets in the range of 80-100%. It also performs significantly well with other classifiers, such as KNN (K=1), SVM (RBF), ET, and DT, while taking the least amount of time across all 18 datasets and selecting comparable features. The performance of the proposed algorithm was also found to be either better than or on par with ten other optimizers with which it was compared. In addition, the single-objective proposed algorithm MDO has been modified to perform optimization for multiple objectives and was tested against nine different multi-objective test problems. The results show that the proposed algorithm performs well and efficiently compared to well-established optimizers like NSGA-II and OMOPSO. The proposals have shown their effectiveness on mathematical and other real-world problems. Due to its efficient search technique, simplicity, and few tunable parameters, it is well-suited for low-cost devices and can be a strong contender for optimizers.