Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 6872162, 13 pages
https://doi.org/10.1155/2022/6872162

*Research Article*

# Improved Hypercube Optimisation Search Algorithm for Optimisation of High Dimensional Functions

**Mustafa Tunay**[1] **and Rahib Abiyev**[2]

[1]*Department of Computer Engineering, Istanbul Gelisim University, Istanbul, Turkey*
[2]*Department of Computer Engineering, Near East University, Lefkosa, Northern Cyprus, Mersin 10, Turkey*

Correspondence should be addressed to Rahib Abiyev; rahib.abiyev@neu.edu.tr

Received 10 September 2021; Revised 28 November 2021; Accepted 10 March 2022; Published 22 April 2022

Academic Editor: Guoqiang Wang



This paper proposes a stochastic search algorithm called improved hypercube optimisation search (HOS+) to find a better solution for optimisation problems. This algorithm is an improvement of the hypercube optimisation algorithm that includes initialization, displacement-shrink and searching area modules. The proposed algorithm has a new random parameters (RP) module that uses two control parameters in order to prevent premature convergence and slow finishing and improve the search accuracy considerable. Many optimisation problems can sometimes cause getting stuck into an interior local optimal solution. HOS+ algorithm that uses a random module can solve this problem and find the global optimal solution. A set of experiments were done in order to test the performance of the algorithm. At first, the performance of the proposed algorithm is tested using low and high dimensional benchmark functions. The simulation results indicated good convergence and much better performance at the lowest of iterations. The HOS+ algorithm is compared with other meta heuristic algorithms using the same benchmark functions on different dimensions. The comparative results indicated the superiority of the HOS+ algorithm in terms of obtaining the best optimal value and accelerating convergence solutions.

## 1. Introduction

The optimisation includes finding the best solutions in a solution space for which the objective function obtains its smallest (or largest) value. Real-world optimisation problems are often nonlinear and can have multiple local optimal (minimum and maximum) solutions. The basic aim is to find the best of these local optimums. Generally, global optimisation includes finding the best available solution from all feasible solutions given in a defined domain for which the objective function will obtain its smallest (or largest) value.

Traditional numerical optimisation algorithms that are based on finding the derivative of the objective function cannot find global optimal points for the function having multiple local optimums. In such cases, one efficient approach is based on the use of heuristic search algorithms. Metaheuristic search algorithms that are based on directed random search methods can provide sufficiently good solutions and solve the local-optimum problem and find global solutions to the optimisation problems [1, 2]. A set of meta-heuristic optimisation algorithms is developed to find the best solutions. These algorithms are: bat algorithm (BAT) [3], cuckoo search (CS) [4], ant lion optimizer (ALO) [5, 6], elephant herding optimisation (EHO) [7, 8], moth-flame optimisation (MFO) [9], krill herd (KH) [10], moth search algorithm (MSA) [11], monarch butterfly optimisation (MBO) [12, 13], mussels wandering optimisation (MWO) [14], and whale optimisation algorithm (WOA) [15]. Other meta-heuristic optimisation algorithms such as differential evolution (DE) [16], biogeography-based optimisation (BBO) [17, 18], harmony search (HS) [19], evolution strategies (ES) [20], sine cosine algorithm (SCA) [21], gravitational search algorithm (GSA) [22], monkey algorithm [23, 24], dragonfly algorithm (DA), and hybrid ABC/DA (HAD) [25, 26] are also efficiently used for solving many optimisation problems.

Paper [27] proposed an intelligent swarm-based MBO algorithm inspired by the migration behavior of monarch butterflies in nature. In this algorithm, the whole population is partitioned into two subpopulations of equal size. Each individual in population 1 changes its position based on the migration operator, each individual in population 2 changes its position according to the butterfly adjusting operator. The algorithm contains exploration and exploitation properties, easy structure, and strong robustness and is designated for global optimisation. The paper [28] proposed a Slime mould algorithm that is based on the oscillation mode of slime mould in nature. The algorithm uses adaptive weights to simulate the process of producing positive and negative feedback of the propagation wave of slime mould based on bio-oscillator to form the optimal path for connecting food with the excellent exploratory ability and exploitation property. In the moth search algorithm [29], the best moth is viewed as a light source. Some neighbour moths that are close to fittest always display an inclination to fly around their own positions in the form of Levy flights. In contrast, the moths that are far from the fittest one will fly towards the best one in a big step. These two operations corresponding to exploration and exploitation are the basis of the MSO algorithm. The paper [30] proposed a population based hunger game search. The algorithm is based on hunger-driven activities and behavioural choice animals. The authors used the algorithm for different areas such as artificial intelligence and machine learning with high optimisation capacity. The paper [31] based on the predation of animals proposed a colony predation search algorithm. The algorithm utilizes mathematical modelling of animal hunting. The algorithm was used for solving engineering problems. The paper [32] proposed a mathematical optimisation model based on simulation of the hunting behavior of Harris hawks. Inspired by the cooperative behavior and chasing style of Harris hawks, the authors designed the algorithm. A number of benchmark examples were used to evaluate the performance of the algorithm.

As mentioned, many metaheuristic optimisation algorithms have been designed to find the best solution to global problems and increase the accuracy of the optimisation. However, the optimisation algorithms can sometimes get stuck into an interior local optimal solution and cannot escape from that state. These search algorithms have premature convergence problems and low search accuracy in solving optimisation problems. This happens due to the loss of diversity among individuals. The original HOS algorithm may also have the same problems and this can lead to a lack of finding a near-optimal solution in the search area. In this paper, we proposed a new version of the HOS algorithm. The novelties of this paper are: The novel structure of the HOS+ algorithm is proposed; The new random perturbation module of the HOS+ algorithm is introduced; The proposed algorithm has been tested on benchmark problems; The proposed HOS+ algorithm help to prevent premature convergence problems and find the best solutions and also improve search accuracy in a small number of iterations. The designed HOS+ algorithm provided passing over possible local optima and has proven to be a successful convergence optima solution for the lowest iterations.

The remainder of the paper is organised as follows: Sec.2 presents the improved HOS+ algorithm. The design stages and operation modules of HOS+ are explained. Sec.3 presents the experimental results and discussion. A set of benchmark functions of different dimensions was used for testing the proposed HOS+ algorithm. In Sec.4 the performance of the HOS+ algorithm is evaluated and compared with the performances of other meta-heuristic algorithms using the same test functions of different dimensions. Finally, the conclusion is presented in Sec 5.

## 2. HOS+ Algorithm

The improved HOS+ algorithm is a new stochastic search method inspired by a hypercube evolution. The algorithm is a derivative-free unconstrained optimisation method and is based on a set of points randomly distributed inside an $m$-dimensional hypercube. The proposed algorithm provides the movement of population (number of points inside of the hypercube) that reaches the minimum (or maximum) of objective function rapidly by reducing the area of the hypercube and updating and searching solutions at each iteration. The original HOS algorithm consists of three-initialization process, displacement- shrink process and searching areas process. The proposed algorithm is renewed by adding a random module in the original search processes.

Stochastic processes are mathematical models of systems that are changing randomly. They are characterised by random variables described by a random probability distribution. They have applications in different fields such as physics, industry, economy, information technology, computer science and many other fields. There are two ways to use a random process in an optimisation problem: through a cost function or a set of constraints. At the same time, stochastic optimisation also refers to any optimisation technique that uses randomness in some ensembles. We consider the case where the parameters of objective function or constraints are random. The improved HOS+ algorithm is inspired by a random process and uses random parameter $p1$ and parameter $p2$ in order to improve the problems of premature convergence and slow finishing and search accuracy considerable. The proposed algorithm is the improvement of the stochastic hypercube optimisation algorithm (HOS) algorithm presented in [1, 2]. The algorithm is presented in Figure 1 and explained by the following steps in detail.

*2.1. Step A: Initialization Process.* The initialization process starts by generating initial points and forms the initial matrices for evaluating solutions in a given hypercube. The initial points are generated using the following operations.

(1) Initialize the solution using the dimension of hypercube.

$$\mathbf{X} = \text{random}\,(n, m), \tag{1}$$

where $m$ is the dimension of the hypercube, $n$ is population size.

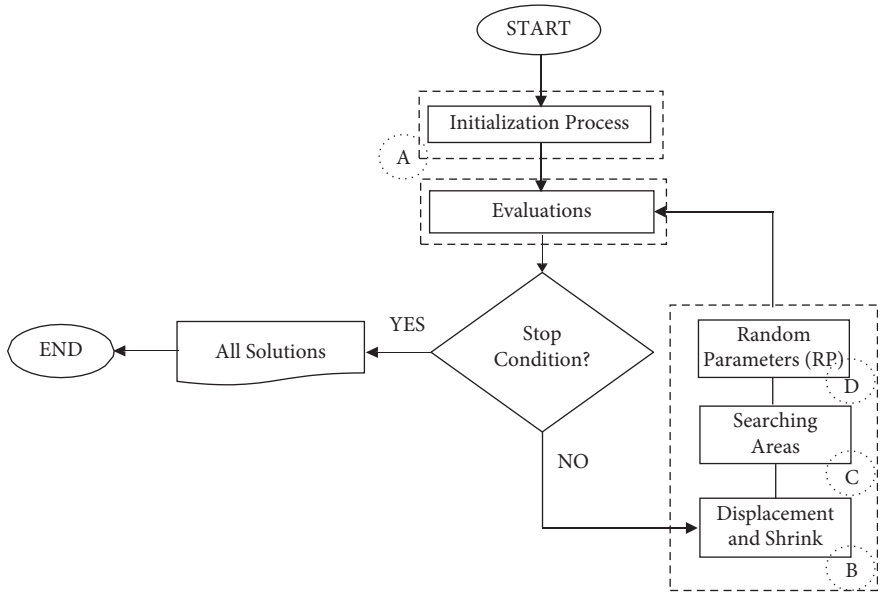


Figure 1: Flowchart of the improved HOS+ algorithm.

(2) Use lower bound (LB) and upper bound (UB) to scale the solutions $x_{ij}$.

$$x_{ij} = \text{LB} + x_{ij}\,(\text{UB} - \text{LB}). \tag{2}$$

(3) Find the radii ($R$) of hypercube.

$$R = \text{UB} - \text{LB}. \tag{3}$$

(4) Find the center of hypercube $X_c$.

$$X_c = R/2, \tag{4}$$

Formula (1) initialize the solutions $X$ inside hypercube which is search area.

$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{1,1} & \cdots & \mathbf{x}_{1,\mathbf{m}} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{\mathbf{n},1} & \cdots & \mathbf{x}_{\mathbf{n},\mathbf{m}} \end{bmatrix}$, where $m$ is the dimension of the hypercube, $n$ is population size. Each position is evaluated using an objective function. The best point $X_{\text{best}}$ is determined according to the values of test (or objective) function **F**.

In the initialization stage, the initial solutions are generated using initial conditions such as the dimension of the hypercube ($m$), radii of the hypercube (**R**), lower-upper boundaries (LB, UB) and a number of points (population, $m$) inside the hypercube (Figure 2). The lower and upper boundaries are used to generate the hypercube. The basic parameters of the hypercube are the center $X_c$ and radii $R$, which are formulated by formulas (3) and (4). In the given search interval, using generated $x_{ij}(i = 1,\ldots,n;\ j = 1,\ldots,m)$ data points inside the hypercube, the values of the objective functions $f_{ij}$ are calculated (here $f_{ij}$ are elements of **F**). After each iteration, the points change their positions (movement). These initial points are evaluated according to the objective function. So initialization process creates matrices as $X_{\text{best}}$, $F_{\text{best}}$ ($nx1$) after evaluating initial points. The

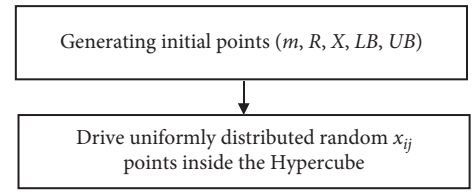


Figure 2: The steps of the initialization process.

determined $X_{best}$ point is improved (updated) using local searches, such as hill climbing or derivative-based local search. If we use a derivative-based local search then $X_{\text{best}}^{\text{new}} = X_{\text{best}} + \rho\nabla F$, where $0 \le p \le 1$, F is the objective function. The details of the initialization process in the HOS+ algorithm are shown in Figure 2. In the next iteration, the $X_{\text{best}}$ is utilised to determine the Hypercube center. This operation is realised by computing the center and mean of the last position point ($X_c$) and the last best $X_{\text{best}}$ point. The given process is called the "displacement" process.

*2.2. Step B: Displacement–Shrink Process.* The displacement–shrink process determines the hypercube's centre and evaluates the test (or objective) function. The centre of the next hypercube is evaluated using the average of the sum of the previous hypercube's centre and the present best point ($X_{\text{best}}$). Thus the centre of the next hypercube (new hypercube) is determined as

$$X_{\text{new\_center}} = \frac{X_c + X_{\text{best}}}{2}, \tag{5}$$

$$R_{\text{new}} = R * S. \tag{6}$$

Here $R$ and $R_{\text{new}}$ are old and new radii, $S$ is the convergence factor calculated in Section 3. The updates of hypercube parameters are performed using (5) and (6). As a

result of this process, the hypercube size and correspondingly the search space are reduced. The process is called "shrink." The decrease in hypercube size leads to an increase in the density of the search points (population). The movement of the best value is governed by contraction. The contraction is greater for smaller movements. This guarantees fast convergence, while it protects against getting stuck at an undesired (local) minimum.

As shown, new data points are generated at each iteration and the objective function is evaluated. According to the evaluation results, the hypercube size is changed. As a result, the hypercube size is decreased and the search space is shrunk correspondingly. The decrease of hypercube size causes an increase in the density of test points. This process causes a rapid finding of the optimum value of the objective function.

The algorithm will pass through a series of points from the current position which determines the maximal distance. The displacement ranges are presented below.

(1) Normalized $x_{ij}$:

$$x_{ij}^{n} = \frac{\left(x_{ij} - X_{c}\right)}{R}. \tag{7}$$

$x_{ij}^{n}$ is a normalized value of $^{ij}_{x}$.

(2) Normalized $X_{\text{best}}$:

$$X_{\text{best}}^{n} = \frac{\left(X_{\text{best}} - X_{c}\right)}{R}. \tag{8}$$

$X_{\text{best}}^{n}$ is the normalized value of $X_{\text{best}}$.

(3) Normalize distance $d_n$:

$$d_{n} = \frac{\left(\text{sum}\left(x_{ij}^{n} - X_{\text{best}}^{n}\right)^{2}\right)^{1/2}}{R}. \tag{9}$$

(4) Re-normalize distance:

$$d_{nn} = \frac{d_{n}}{\sqrt{m}}. \tag{10}$$

The $x$ displacement is calculated and normalized twice for each iteration: at first, each element of $x$ is divided by its corresponding initial interval so that the displacement is converted into unity-sided points (equations (7) and (8)), and then this number is again normalized by dividing it to the diagonal of the points such as $\sqrt{m}$ (equations (9) and (10)). Thus, the contraction of the hypercube is becoming higher, when the movement of the number of points shrink which accelerates the convergence.

*2.3. Step C: Searching Areas Process.* Using equations (7)–(10) the distances between new and old optimum values are calculated in this process. In addition, the "Searching areas" process uses the interval defined for re-normalized distance, given in (11), to control the movements of $x$.

$$0 \le dm \le 1. \tag{11}$$

In case of satisfaction the condition by the movement $x$, the convergence factor $S$ is computed and updated at each iteration as

$$S = 1 - 0.2e^{-3d_{nn}}. \tag{12}$$

In the above equation, $d_{nn}$ is the normalized distance calculated by (10) and based on the average of the last two best values of $x$. Thus, the purpose of the proposed algorithm ensures the movement of the population that reaches the minimum point rapidly by reducing the area of the hypercube after each iteration. A flowchart of the searching areas process in the proposed algorithm is shown in Figure 3.

*2.4. Step D: Random Parameter (RP) Module.* HOS+ algorithm includes a new RP module characterised by two control parameters *p1* and *p2*. This module improves the points (current positions) inside the hypercube that might get stuck at some local solutions. At first, the *p1* improves the points having local optima problem. The process is continued according to some tolerance and fixed by *tolX*. In addition, the upper bound of dimension $d$ can be determined according to the value of *tolX*. The value of the solution at the local point is updated by multiplying the parameter p1 by a random scalar drawn from the standard normal distribution, that is $X^{\text{new}} \leftarrow X*(1 + p1*\text{rand}n)$. Here rand$n$ generates random numbers in the interval of 0 and 1. The new point will be accepted according to the values of test functions. If the value of the optimisation function in the new point will be minimum (or maximum) than the previous one then the new point will be included in the solution. Thus, by using this operation the presented random module prevents the point from getting stuck in some local optimal solutions while controlling the points' positions inside the hypercube. After these operations, the second new random parameter is introduced in order to control directed movements of all points inside the hypercube. The introduced second parameter ($p2$) improves the solutions along the direction pointing to their current position with different perturbations originating from some possible local minimum and searching for another minimum point. The points are updated by multiplying the parameter p2 to uniformly distributed random numbers (rand [1 x $D$]). That is

$$X^{\text{new}}(\cdot) \leftarrow X(\cdot) * (1 + p2 * \text{rand}[1, D]). \tag{13}$$

where $D$ is searching dimension. Thus, the improvements of positions are performed along the direction pointing with different perturbations in order to exit from some possible local minimum or to search for another minimum point. The pseudocode of the random permutation module is given in Figure 4.

The computational complexity of the HOS+ algorithm was analyzed. HOS+ includes initialization, fitness evaluation, displacement-shrink and normalization, searching areas and random parameter modules. Hypercube dimensions $n$, population size $m$ and a maximum number of iterations $T$ are the main parameters affecting the running time of the HOS+ algorithm in these modules.

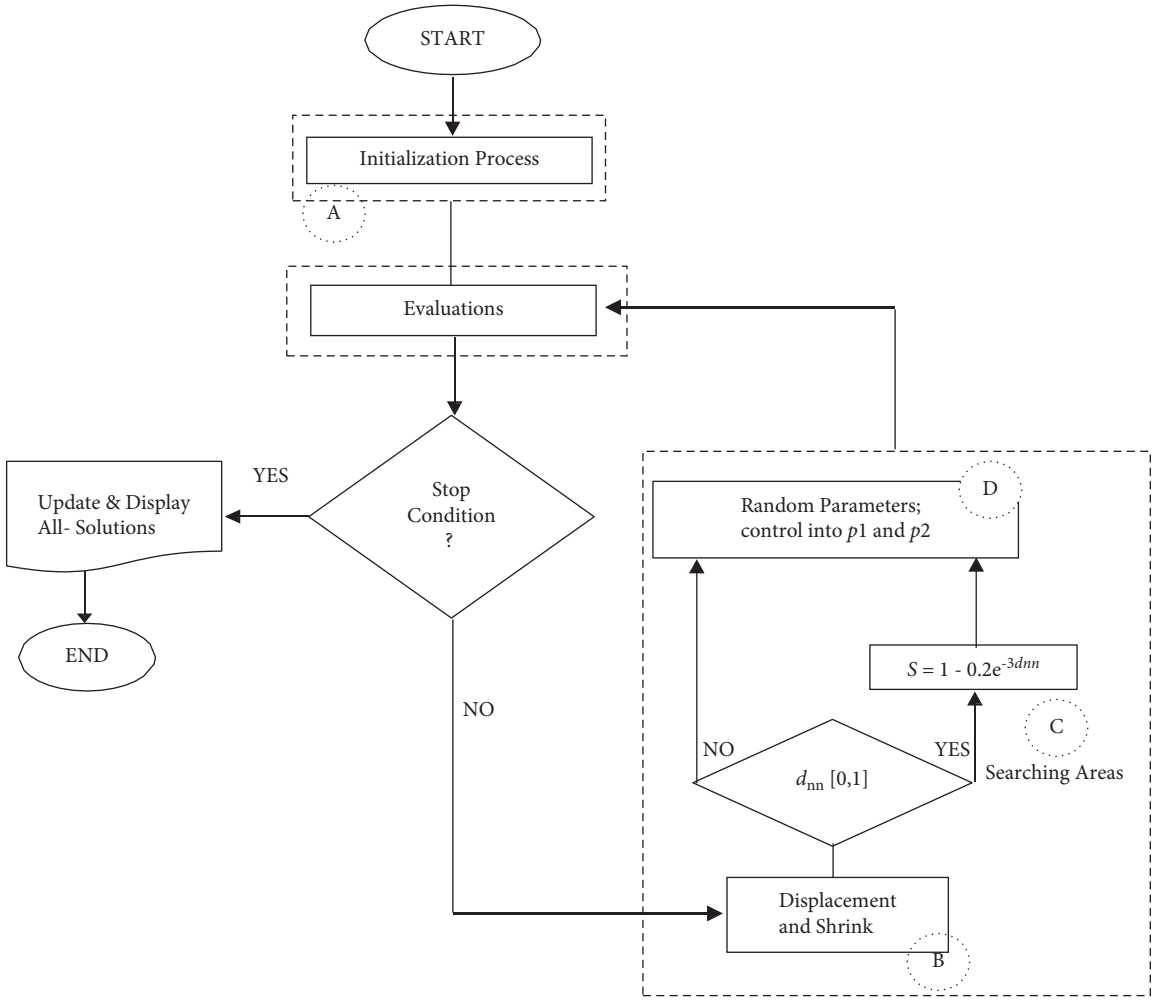


Figure 3: Flow chart of the searching areas process in the proposed algorithm.

Computational complexity of initialization is $O(n{\cdot}m)$, displacement- shrink and normalization module is $O(n{\cdot}m)$, searching areas $O(n)$, random permutation module is $O(n{\cdot}m)$. The displacement-shrink and normalization module, searching areas, random permutation module running in each iteration $t$; if we take into account the maximum number of iterations $T$ then the computational complexity of HOS+ will be presented by $O(n{\cdot}m + n{\cdot}T + n{\cdot}m{\cdot}T)$.

## 3. Benchmark Functions

The benchmark functions used are Sphere function (F1), Schwefel 2.22 function (F2), Rotated Hyper-Ellipsoid function (F3), Ackley function (F4), Griewank function (F5), and Hyperellipsoid function (F6). The details of information for these test functions are given below. The performance of the HOS+ was evaluated using low, medium and high dimensional optimisation functions. In the paper, low dimension is taken equal to 30D, medium dimension equal to 60D and high dimension-90D. For more information about these benchmark functions, we refer the reader to the link: https://www.sfu.ca/~ssurjano/optimisation.html.

*3.1. Sphere Function (F1).* The function is convex, continuous, differentiable, separable and uni-modal. It is used $x_i \in [-5.12, 5.12]$ for all $i = 1,\ldots,n$ and the global minimum is at $f(x) = 0$.

$$f(x) = f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} x_i^2. \tag{14}$$

*3.2. Schwefel 2.22 Function (F2).* The function is convex, continuous, non-differentiable, separable and uni-modal. It is used $x_i \in [-10,10]$ for all $i = 1,\ldots,n$ and the global minimum is at $f(x) = 0$.

$$f(x) = f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|. \tag{15}$$

*3.3. Rotated Hyper Ellipsoid Function (F3).* This function is convex, continuous and uni-modal. It is used $x_i \in [-65, 65]$ for all $i = 1,\ldots,n$ and the global minimum is at $f(x) = 0$.

```
An RP module (→ Step D) of the HOS+ algorithm
Input (Step D): Number of Population (n), Number of Tolerance (tolX), Lower
Bound (LB) and Upper Bound (UB), Random Parameters (p1 and p2), randn [0,1],
Global maximum is searched (min)
Output: Best Solution (Best (X))
for i = 1 to n do
    for j = 1 to tolX do
        d ← ceil (rand x D)
        X_i^new ← X_id * (1 + p1 * randn)
        if LB < X_i^new < UB then
            Continue
        end if
        if min > 0 then
        | if f(X_i^new) > f(X_ij) then
            | X_ij ← X_i^new
        | end if
        Else
        | if f(X_i^new) < f(X_ij) then
        |   | X_ij ← X_i^new
        | end if
        end if
    end for
end for
for i = 1 to n do
    for j = 1 to tolX do
        X_i^new ← X_ij (·) (1 + p2 * rand [1, D])
        if LB < X_i^new < UB then
            Continue
        end if
        if min > 0 then
        | if f(X_i^new) > f(X_ij) then
        |   | X_ij ← X_i^new
        | end if
        Else
        | if f(X_i^new) < f(X_ij) then
        |   | X_ij ← X_i^new
        | end if
        end if
    end for
end for
Output
```

Figure 4: Pseudocode of random perturbation module.

$$f(x) = f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2. \tag{16}$$

*3.4. Ackley Function (F4).* This function is continuous and multi-modal. It is used $x_i \in [-32, 32]$ for all $i = 1,\ldots,n$ and the global minimum is at $f(x) = 0$.

$$f(x) = f(x_1, x_2, \ldots, x_n) = -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e. \tag{17}$$

*3.5. Griewank Function (F5).* This function is continuous and uni-modal. It is used $x_i \in [-600, 600]$ for all $i = 1,\ldots,n$ and the global minimum is at $f(x) = 0$.

$$f(x) = f(x_1, x_2, \ldots, x_n) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right). \tag{18}$$

*3.6. Hyper Ellipsoid Function (F6).* This function is convex, continuous, differentiable, separable and uni-modal. It is used $x_i \in [-5.12, 5.12]$ for all $i = 1,\ldots,n$ and the global minimum is at $f(x) = 0$.

$$f(x) = f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} i^2 x_i^2. \tag{19}$$

Table 1: The performances of the HOS and HOS+ algorithms using different dimensions.

| Benchmark functions | $D$ | HOS Best | HOS Mean | HOS Std. Dev. | HOS+ Best | HOS+ Mean | HOS+ Std. dev. |
|---|---|---|---|---|---|---|---|
| Sphere (F1) | 30 | $1.19E-06$ | $8.88E+00$ | $4.86E+00$ | $9.95E-137$ | $2.63E-51$ | $4.29E-51$ |
| | 60 | $1.30E-02$ | $5.92E+01$ | $4.00E+01$ | $7.79E-96$ | $6.39E-39$ | $6.19E-39$ |
| | 90 | $1.20E-01$ | $1.49E+02$ | $7.49E+01$ | $1.70E-81$ | $2.19E-32$ | $3.59E-32$ |
| Schwefel 2.22 (F2) | 30 | $3.00E-01$ | $2.67E+01$ | $1.41E+01$ | $1.19E-95$ | $1.98E-27$ | $3.19E-27$ |
| | 60 | $6.50E-01$ | $1.01E+02$ | $4.40E+01$ | $3.21E-72$ | $3.88E-22$ | $6.39E-22$ |
| | 90 | $4.75E+00$ | $1.67E+02$ | $7.66E+01$ | $2.49E-59$ | $5.72E-18$ | $6.91E-18$ |
| Rotated hyper ellipsoid (F3) | 30 | $2.59E+02$ | $2.10E+04$ | $1.11E+04$ | $3.39E-134$ | $2.69E-47$ | $2.03E-47$ |
| | 60 | $4.08E+02$ | $2.77E+05$ | $1.59E+05$ | $5.25E-109$ | $1.75E-34$ | $2.09E-34$ |
| | 90 | $5.85E+02$ | $9.81E+05$ | $5.09E+05$ | $1.05E-75$ | $1.35E-28$ | $1.79E-28$ |
| Ackley (F4) | 30 | $5.00E-03$ | $1.06E+01$ | $3.49E+00$ | $8.88E-16$ | $3.57E-15$ | $4.69E-15$ |
| | 60 | $5.40E-01$ | $1.51E+01$ | $3.86E+00$ | $8.88E-16$ | $2.53E-14$ | $2.79E-14$ |
| | 90 | $5.70E-01$ | $1.69E+01$ | $4.40E+00$ | $4.44E-15$ | $4.29E-13$ | $6.69E-13$ |
| Griewank (F5) | 30 | $5.00E-02$ | $3.10E+01$ | $1.85E+01$ | $0.00E+00$ | $0.00E+00$ | $0.00E+00$ |
| | 60 | $6.90E-01$ | $1.50E+02$ | $1.19E+02$ | $0.00E+00$ | $0.00E+00$ | $0.00E+00$ |
| | 90 | $1.20E+01$ | $4.19E+02$ | $2.39E+02$ | $0.00E+00$ | $0.00E+00$ | $0.00E+00$ |
| Hyper ellipsoid (F6) sphere (F1) | 30 | $5.10E+01$ | $2.61E+02$ | $4.59E+01$ | $1.55E-139$ | $2.19E-48$ | $1.77E-48$ |
| | 60 | $6.30E+01$ | $2.75E+03$ | $6.49E+02$ | $4.35E-113$ | $3.39E-36$ | $3.59E-36$ |
| | 90 | $6.90E+01$ | $1.04E+04$ | $2.45E+03$ | $4.89E-79$ | $2.99E-29$ | $3.96E-29$ |

## 4. The Performance of HOS+ Algorithm on Benchmark Functions

The HOS+ algorithm is simulated in Matlab R2017a for finding optimal solutions for a set of benchmark functions. The computer used for simulations has the following characteristics;

(i) CPU: i5-8250U

(ii) CPU Speed:1.60 GHz–1.80 GHz

(iii) RAM: 8.00 GB

(iv) OS: Windows 10

The HOS+ algorithm has been tested using the above-mentioned benchmark functions on 30D, 60D, and 90D dimensions. Evaluations are carried out using the same population size of 50, the same number of iterations of 50 and the maximum function evaluation. For all cases, the results are averaged using 100 independent runs of the algorithm. For measuring the performances of the algorithm the best, mean and standard deviation are taken.

At first, the performance of the HOS+ algorithm is compared with the original HOS algorithm given in [1, 2]. Using both algorithms the experiments were conducted for all benchmark functions on 30, 60 and 90 dimensions. Table 1 depicts the results of experiments obtained for six optimisation functions of F1, F2, F3, F4, F5, and F6 for the HOS and HOS+ algorithms. The simulations have been done at the same initial conditions. The best, averaged values of mean and standard deviation are illustrated in the table. The convergence plots and time-spent of HOS+ algorithm obtained from the simulations on 90-dimensional optimisation functions F1, F2, F3, F4, F5, and F6 were depicted in Figures 5 to 10, correspondingly. The results of the experiments were presented using the convergence plots and global search ability of the proposed algorithm. For comparative purpose, the convergence plots of original HOS algorithm are presented in Figure 11. The comparative results of performances given in Table 1 and the convergence plots given in Figures 5 to 10 and Figure 11 demonstrate the superiority of the HOS+ algorithm over the original HOS algorithm.

## 5. Comparison of HOS+ with Other Metaheuristic Algorithms

The HOS+ algorithm performance was compared with the performances of other meta-heuristic optimisation algorithms using 6 test functions on different dimensions, particularly 30D, 60D, and 90D. The comparisons of the algorithms were done using the same initial conditions. All algorithms are simulated using the same iterations' number, the same dimensions, and the same maximum function evaluation [26, 33].

The comparative results of each function are presented in Tables 2–5. The best results are marked in bold. Tables 2–4 depict comparative results of experiments obtained for optimisation functions (F1–F6) on dimensions 30, 60 and 90.100 independent runs have been done for each optimisation function using HOS+ algorithm. Table 5 demonstrates the experimental comparative results of the F1, F2, and F4 functions on dimensions 20, 50, and 100. The results are averaged values of 30 independent runs of each algorithm.

The initial values of the parameters for the HOS+ algorithm were set as follows: population size is set equal to 50 and a number of iterations is set equal to 50.

First, the proposed algorithm was compared with a selected collection of other meta-heuristic algorithms. DA, ABC, and HAD algorithms were taken for comparison. Table 2 illustrates the best, mean and best standard deviation obtained from the experiments.

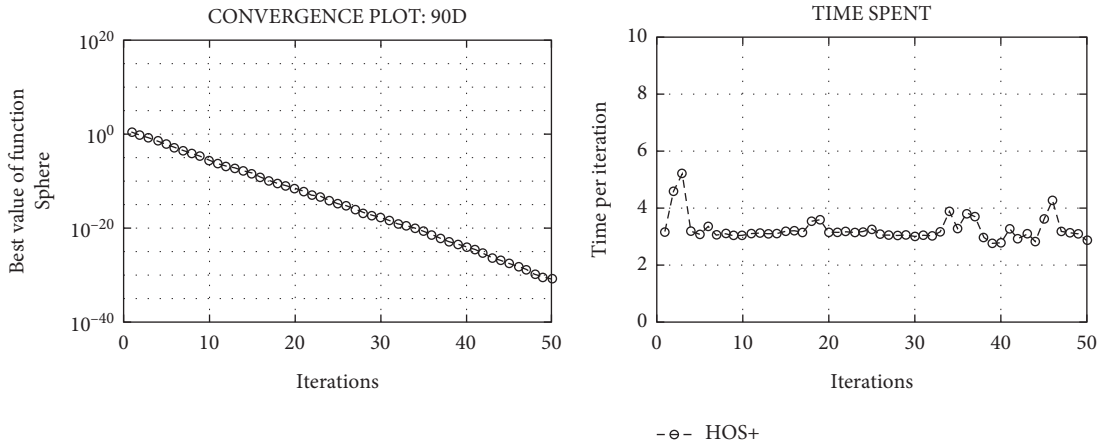


Figure 5: Convergence plot with time spent against 90-dimensional Sphere function (F1).

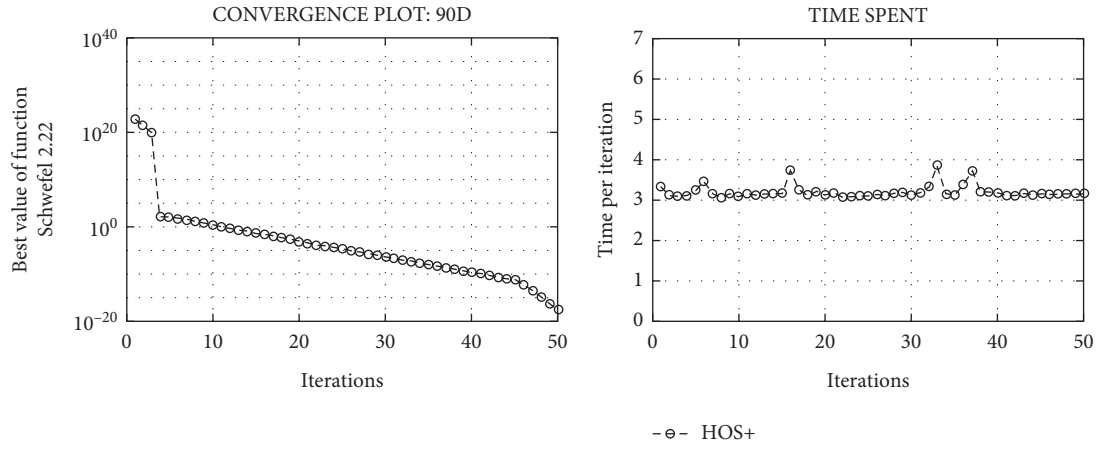


Figure 6: Convergence plot with time spent against 90-dimensional Schwefel 2.22 function (F2).

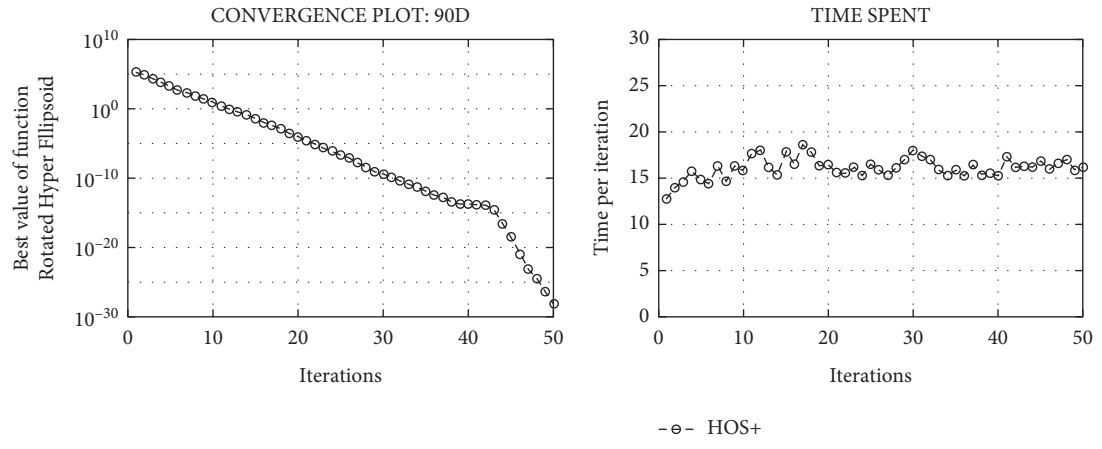


Figure 7: Convergence plot with time spent against 90-dimensional rotated Hyperellipsoid function (F3).

In the second stage, the HOS+ algorithm was compared with the meta-heuristic optimisation algorithms ACO, GA, DE, and PSO. Table 3 depicts the experimental results for each function.

In the third stage, the HOS+ algorithm was compared with the EHO, MSA, and WOA meta-heuristic optimisation algorithms. Table 4 depicts the experimental results obtained for each function.

In the fourth stage, the HOS+ algorithm was compared with the monarch butterfly optimisation algorithm (MBO), MBO with opposition-based learning and random local perturbation (OPMBO), and MBO with greedy

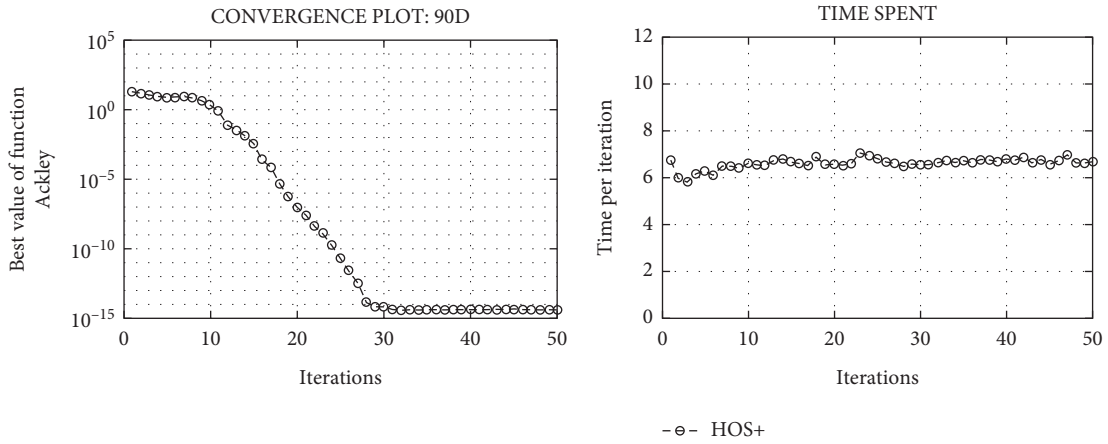


Figure 8: Convergence plot with time spent against 90-dimensional ackley function (F4).

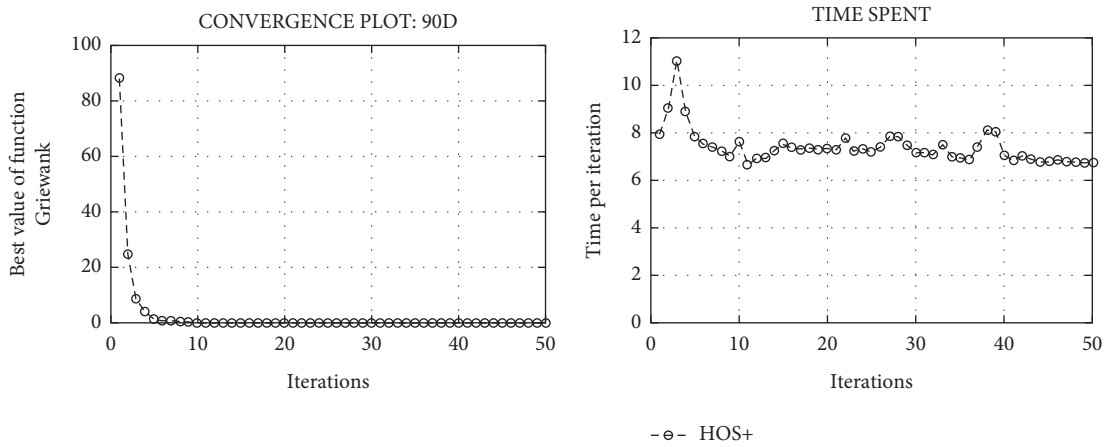


Figure 9: Convergence plot with time spent against 90-dimensional Griewank function (F5).

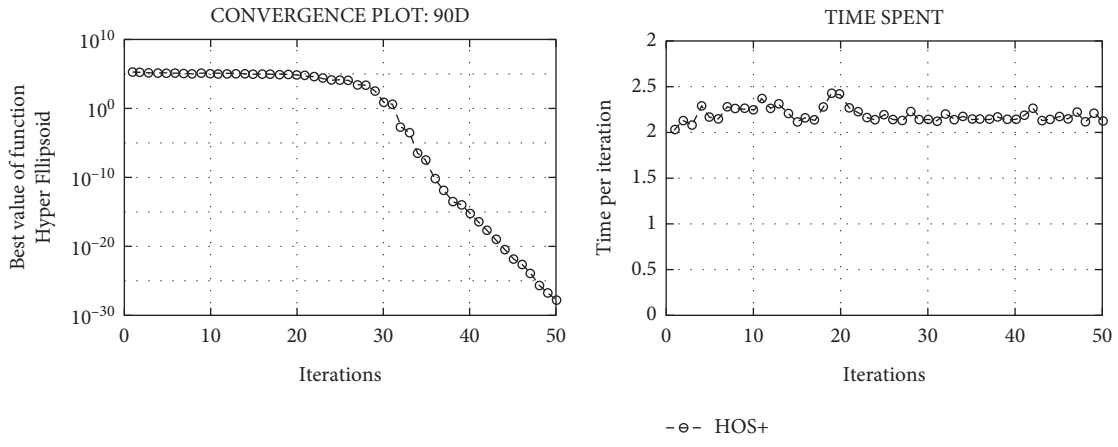


Figure 10: Convergence plot with time spent against 90-dimensional Hyperellipsoid function (F6).

strategy and self-adaptive crossover operator (GCMBO) using three benchmark functions. The comparative results are presented in Table 5. The best results are marked in bold.

In the fifth stage, the simulation results of HOS+ algorithm is compared with the simulation results of sine-cosine algorithm (SCA), m-SCA [34] and improved crow search algorithm ICSA [35]. Table 6 shows the experimental comparative results of the F1, F2, F3, F4 and F5 functions on 30 dimensions.

The experimental comparative results of ACO, ABC, DA DE, HAD, GA, PSO, EHO, MSA, WOA, MBO, GCMBO, OPMBO, ICSA, SCA, M-SCA and HOS+ algorithm showed

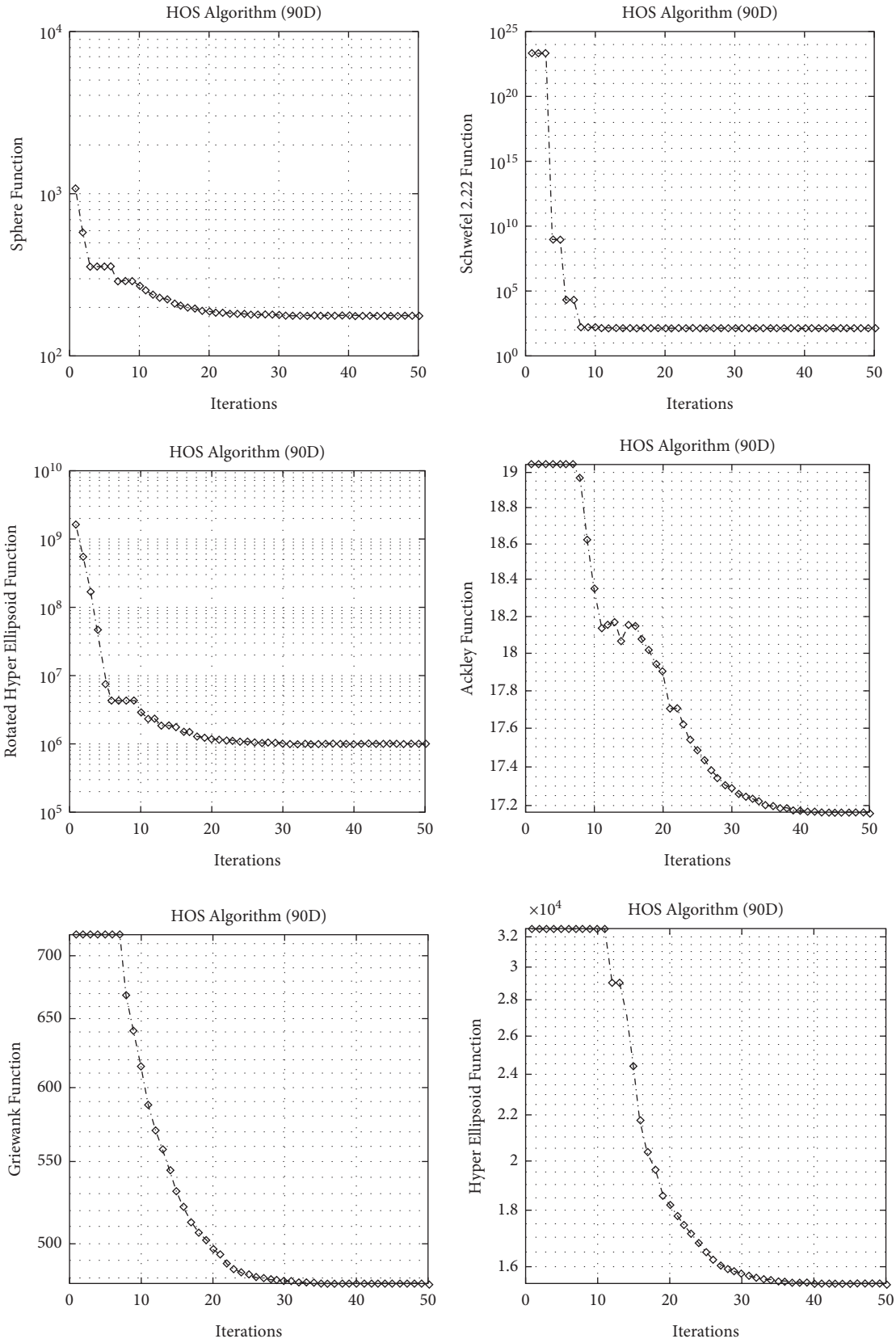


Figure 11: Convergence plots of original HOS algorithm for optimisation functions (F1–F6).

Table 2: The mean of test functions values found by ABC, DA, HAD, and HOS+

| F | $D$ | ABC | DA | HAD | HOS+ |
|---|---|---|---|---|---|
| F1 | 30 | $2.98E+01$ | $1.76E+00$ | $2.61E-14$ | $2.63E-51$ |
| | 60 | $1.97E+02$ | $2.29E+00$ | $3.46E-10$ | $6.39E-39$ |
| | 90 | $4.07E+02$ | $8.53E+00$ | $2.05E-09$ | $2.19E-32$ |
| F2 | 30 | $7.54E+00$ | $7.46E+00$ | $3.40E-08$ | $1.98E-27$ |
| | 60 | $6.19E+01$ | $2.80E+01$ | $4.66E-06$ | $3.88E-22$ |
| | 90 | $1.59E+02$ | $3.10E+01$ | $4.89E-05$ | $5.72E-18$ |
| F3 | 30 | $4.85E+04$ | $1.96E+03$ | $9.76E-14$ | $2.69E-47$ |
| | 60 | $9.03E+05$ | $2.70E+04$ | $2.56E-08$ | $1.75E-34$ |
| | 90 | $2.92E+06$ | $1.53E+05$ | $1.24E-07$ | $1.35E-28$ |
| F4 | 30 | $1.70E+01$ | $1.76E+00$ | $5.89E-08$ | $3.57E-15$ |
| | 60 | $1.95E+01$ | $7.41E+00$ | $7.18E-06$ | $2.53E-14$ |
| | 90 | $2.00E+01$ | $7.00E+00$ | $3.08E-05$ | $4.29E-13$ |
| F5 | 30 | $9.22E+01$ | $7.64E+00$ | $4.79E-12$ | $0.00E+00$ |
| | 60 | $6.93E+02$ | $1.74E+01$ | $8.68E-09$ | $0.00E+00$ |
| | 90 | $1.43E+03$ | $2.27E+01$ | $2.72E-07$ | $0.00E+00$ |
| F6 | 30 | $2.36E+02$ | $4.98E+00$ | $5.16E-14$ | $2.19E-48$ |
| | 60 | $7.72E+03$ | $1.41E+02$ | $1.16E-10$ | $3.39E-36$ |
| | 90 | $3.85E+04$ | $6.56E+02$ | $2.09E-08$ | $2.99E-29$ |

Table 3: The mean of test functions values found by ACO, DE, GA, PSO, and HOS+.

| F | $D$ | ACO | DE | GA | PSO | HOS+ |
|---|---|---|---|---|---|---|
| F1 | 30 | $1.63E+02$ | $2.79E+01$ | $9.58E+01$ | $5.12E+01$ | $2.63E-51$ |
| | 60 | $3.76E+02$ | $1.74E+02$ | $2.86E+02$ | $2.13E+02$ | $6.39E-39$ |
| | 90 | $6.02E+02$ | $3.80E+02$ | $4.65E+02$ | $4.29E+02$ | $2.19E-32$ |
| F2 | 30 | $1.13E+02$ | $5.38E+01$ | $8.60E+01$ | $1.14E+02$ | $1.98E-27$ |
| | 60 | $2.48E+02$ | $1.71E+02$ | $2.03E+02$ | $2.49E+02$ | $3.88E-22$ |
| | 90 | $3.88E+02$ | $2.97E+02$ | $3.23E+02$ | $3.89E+02$ | $5.72E-18$ |
| F3 | 30 | $2.13E+05$ | $5.52E+04$ | $1.41E+05$ | $9.79E+04$ | $2.69E-47$ |
| | 60 | $1.65E+06$ | $7.03E+05$ | $1.18E+06$ | $9.92E+05$ | $1.75E-34$ |
| | 90 | $4.30E+06$ | $2.93E+06$ | $3.20E+06$ | $3.47E+06$ | $1.35E-28$ |
| F4 | 30 | $1.85E+01$ | $1.87E+01$ | $1.77E+01$ | $1.87E+01$ | $3.57E-15$ |
| | 60 | $1.90E+01$ | $1.90E+01$ | $1.86E+01$ | $1.90E+01$ | $2.53E-14$ |
| | 90 | $1.91E+01$ | $1.91E+01$ | $1.88E+01$ | $1.91E+01$ | $4.29E-13$ |
| F5 | 30 | $8.57E+01$ | $9.38E+01$ | $1.27E+02$ | $1.69E+02$ | $0.00E+00$ |
| | 60 | $4.32E+02$ | $6.02E+02$ | $4.64E+02$ | $7.27E+02$ | $0.00E+00$ |
| | 90 | $7.13E+02$ | $1.31E+03$ | $8.87E+02$ | $1.62E+03$ | $0.00E+00$ |
| F6 | 30 | $2.34E+03$ | $1.75E+02$ | $1.20E+02$ | $2.79E+02$ | $2.19E-48$ |
| | 60 | $2.28E+04$ | $4.88E+03$ | $7.93E+03$ | $3.44E+03$ | $3.39E-36$ |
| | 90 | $8.42E+04$ | $2.61E+04$ | $4.28E+04$ | $1.48E+04$ | $2.99E-29$ |

Table 4: The mean of test functions values found by EHO, MSA, WOA, and HOS+.

| F | $D$ | EHO | MSA | WOA | HOS+ |
|---|---|---|---|---|---|
| F1 | 30 | $2.49E-07$ | $2.30E-08$ | $2.42E-09$ | $2.63E-51$ |
| | 60 | $6.44E-07$ | $3.67E-07$ | $4.67E-09$ | $6.39E-39$ |
| | 90 | $1.06E-06$ | $1.17E-06$ | $7.45E-09$ | $2.19E-32$ |
| F2 | 30 | $4.12E-03$ | $1.78E-04$ | $3.76E-05$ | $1.98E-27$ |
| | 60 | $9.34E-03$ | $9.50E-04$ | $9.19E-05$ | $3.88E-22$ |
| | 90 | $1.46E-02$ | $1.63E-03$ | $1.52E-04$ | $5.72E-18$ |
| F3 | 30 | $4.59E-04$ | $3.02E-07$ | $6.47E-06$ | $2.69E-47$ |
| | 60 | $2.41E-03$ | $8.01E-06$ | $2.97E-05$ | $1.75E-34$ |
| | 90 | $6.11E-03$ | $4.79E-05$ | $6.02E-05$ | $1.35E-28$ |
| F4 | 30 | $1.94E-03$ | $6.84E-05$ | $1.21E-04$ | $3.57E-15$ |
| | 60 | $2.22E-03$ | $1.88E-04$ | $1.19E-04$ | $2.53E-14$ |
| | 90 | $2.33E-03$ | $3.57E-04$ | $1.41E-04$ | $4.29E-13$ |

Table 4: Continued.

| F | $D$ | EHO | MSA | WOA | HOS+ |
|---|---|---|---|---|---|
| F5 | 30 | $1.44E-04$ | $1.09E-09$ | $6.10E-02$ | $0.00E+00$ |
| | 60 | $2.14E-04$ | $7.15E-09$ | $3.88E-02$ | $0.00E+00$ |
| | 90 | $2.58E-04$ | $3.90E-08$ | $3.47E-02$ | $0.00E+00$ |
| F6 | 30 | $4.21E-06$ | $3.33E-06$ | $2.85E-08$ | $2.19E-48$ |
| | 60 | $4.57E-05$ | $1.44E-04$ | $3.31E-07$ | $3.39E-36$ |
| | 90 | $1.82E-04$ | $9.07E-04$ | $1.89E-06$ | $2.99E-29$ |

Table 5: The mean of test functions values found by MBO, GCMBO, OPMBO, and HOS+.

| F | $D$ | MBO | GCMBO | OPMBO | HOS+ |
|---|---|---|---|---|---|
| F1 | 20 | $1.02E+01$ | $4.03E-09$ | $1.99E-10$ | $4.69E-55$ |
| | 50 | $2.09E+02$ | $1.11E-09$ | $8.85E-09$ | $2.99E-44$ |
| | 100 | $4.83E+02$ | $1.00E+01$ | $4.87E+00$ | $1.20E-30$ |
| F2 | 20 | $1.95E+01$ | $2.27E+00$ | $1.72E-06$ | $4.68E-33$ |
| | 50 | $1.30E+02$ | $2.36E+01$ | $2.73E-05$ | $1.69E-26$ |
| | 100 | $2.66E+02$ | $5.61E+01$ | $7.60E-02$ | $3.77E-16$ |
| F4 | 20 | $8.92E+00$ | $2.61E-01$ | $6.94E-06$ | $1.29E-16$ |
| | 50 | $1.59E+01$ | $1.96E+00$ | $4.03E-05$ | $1.59E-15$ |
| | 100 | $1.86E+01$ | $8.63E+00$ | $3.93E-02$ | $3.69E-12$ |

Table 6: The mean of test functions values found by ICSA, SCA, M-SCA and HOS+ algorithms with $D=30$.

| | F1 | F2 | F3 | F4 | F5 |
|---|---|---|---|---|---|
| ICSA | $3.06E-5$ | $8.15E-6$ | $8.21E+2$ | $2.80E-3$ | $3.49E-2$ |
| SCA | $1.87E+01$ | $1.79E-02$ | $8.84E+03$ | $1.83E+01$ | $9.38E-01$ |
| M-SCA | $5.70E-03$ | $9.11E-04$ | $8.48E+02$ | $3.36E-03$ | $3.84E-02$ |
| HOS+ | $2.63E-51$ | $1.98E-27$ | $2.69E-47$ | $3.57E-15$ | $0.00E+00$ |

that the proposed HOS+ algorithm has obtained better results and best convergence due to escaping local optimums in the majority of the evaluations. The obtained simulation results indicate the effectiveness of using HOS+ algorithm in optimisation problems.

## 6. Conclusions

This paper proposes a novel stochastic search algorithm based on the evolution of hypercube. The design stages of the algorithm were explained. A new random perturbation module is introduced in order to solve local optimum problems in optimisation problems and to find a global solution. The HOS+ algorithm has been tested using various low and high dimensional optimisation functions and the solution of the specified local optimum problem has been proven by experimental results. The obtained results demonstrated that the algorithm can successfully avoid getting stuck in the local optimum and find a global solution for the lowest iterations. Comparative results of performances that include the best, the mean, standard deviation and convergence plots demonstrate advantages of the proposed HOS+ algorithm over other thirteen meta-heuristic algorithms. The obtained simulation results indicate the efficiency of using the HOS+ algorithm in the solution of optimisation problems. Future research includes the application of the HOS+ algorithm to solve practical optimisation problems.

## Data Availability

No data were used to support this study.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] R. H. Abiyev and M. Tunay, "Optimization of high dimensional functions through hypercube evaluation," *Computational Intelligence and Neuroscience*, vol. 2015, Article ID 967320, 11 pages, 2015.

[2] R. H. Abiyev and M. Tunay, "Optimization search using hypercubes," in *Proceedings of the 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 1–8, Istanbul, Turkey, October 2020.

[3] S. Srivastava and S. K. Sahana, "Application of bat algorithm for transport network design problem," *Applied Computational Intelligence and Soft Computing*, vol. 2019, no. 5, pp. 1–12, Article ID 9864090, 2019.

[4] M. Mareli and B. Twala, "An adaptive Cuckoo search algorithm for optimisation," *Applied Computing and Informatics*, vol. 14, no. 2, pp. 107–115, 2018.

[5] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015.

[6] K. Roy, K. K. Mandal, and A. C. Mandal, "Ant-Lion Optimizer algorithm and recurrent neural network for energy

management of micro grid connected system," *Energy*, vol. 167, pp. 402–416, 2019.

[7] G.-G. Wang, S. Deb, and L. S. Coelho, "Elephant herding optimization," in *Proceedings of the 3rd International Symposium on Computational and Business Intelligence (ISCBI)*, pp. 1–5, Bali, Indonesia, December 2015.

[8] J. Li, H. Lei, A. H. Alavi, and G.-G. Wang, "Elephant herding optimization: variants, hybrids, and applications," *Mathematics*, vol. 8, no. 9, 2020.

[9] Y. Li, X. Zhu, and J. Liu, "An improved moth-flame optimization algorithm for engineering problems," *Symmetry*, vol. 12, no. 8, 2020.

[10] G.-G. Wang and A. H. Gandomi, "A comprehensive review of krill herd algorithm: variants, hybrids and applications," *Artificial Intelligence Review*, vol. 51, pp. 119–148, 2019.

[11] I. Strumberger and N. Bacanin, "Modified moth search algorithm for global optimization problems," *International Journal of Computer*, vol. 3, pp. 44–48, 2018.

[12] H. Hu, Z. Cai, S. Hu, Y. Cai, J. Chen, and S. Huang, "Improving monarch butterfly optimization algorithm with self-adaptive population," *Algorithms*, vol. 11, no. 5, 2018.

[13] Y. Feng, S. Deb, G. G. Wang, and A. H. Alavi, "Monarch butterfly optimization: a comprehensive review," *Expert Systems with Applications*, vol. 168, Article ID 114418, 2021.

[14] A. Abusnaina, R. Abdullah, and A. Kattan, "Self-adaptive mussels wandering optimization algorithm with application for artificial neural network training," *Journal of Intelligent Systems*, vol. 29, no. 1, pp. 345–363, 2020.

[15] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[16] M. Georgioudakis and V. A. Plevris, "Comparative study of differential evolution variants in constrained structural optimization," *Frontiers in Built Environment*, vol. 6, 2020.

[17] U. Guvenc, A. Isık, T. Yigit, and I. Akkaya, "Performance analysis of biogeography-based optimization for automatic voltage regulator system," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 24, no. 3, pp. 1150–1162, 2016.

[18] H. Garg, "An efficient biogeography based optimization algorithm for solving reliability optimization problems," *Swarm and Evolutionary Computation*, vol. 4, pp. 1–10, 2015.

[19] X. Z. Gao, V. Govindasamy, H. Xu, X. Wang, and K. Zenger, "Harmony search method: theory and applications," *Computational Intelligence and Neuroscience*, vol. 2015, Article ID 258491, 10 pages, 2015.

[20] Z. Li, X. Lin, Q. Zhang, and H. Liu, "Evolution strategies for continuous optimization: a survey of the state-of-the-art," *Swarm and Evolutionary Computation*, vol. 56, Article ID 100694, 2020.

[21] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[22] E. Rashedi, E. Rashedi, and H. Nezamabadi-pour, "A comprehensive survey on gravitational search algorithm," *Swarm and Evolutionary Computation*, vol. 41, pp. 141–158, 2018.

[23] R. H. Abiyev and M. Tunay, "Experimental study of specific benchmarking functions for modified monkey algorithm," *Procedia Computer Science*, vol. 102, pp. 595–602, 2016.

[24] M. Tunay, "A new design of metaheuristic search called improved monkey algorithm based on random perturbation for optimization problems," *Scientific Programming*, vol. 2021, Article ID 5557259, 14 pages, 2021.

[25] C. Rahman and T. Rashid, "A survey on dragonfly algorithm and its applications in engineering," *Journal of Computational Design and Engineering*, vol. 2020, pp. 1–23, 2020.

[26] W. A. H. M. Ghanem and A. Jantan, "A cognitively inspired hybridization of artificial bee colony and dragonfly algorithms for training multi-layer perceptrons," *Cognitive Computation*, vol. 10, pp. 1096–1134, 2018.

[27] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Computing & Applications*, vol. 31, pp. 1–20, 2015.

[28] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: a new method for stochastic optimization," *Future Generation Computer Systems*, vol. 111, pp. 300–323, 2020.

[29] G.-G. Wang, "Moth search algorithm: a bioinspired metaheuristic algorithm for global optimization problems," *Memetic Computing*, vol. 10, pp. 151–164, 2016.

[30] Y. Yang, H. Chen, A. A. Heidari, and A. HGandomi, "Hunger games search: visions, conception, implementation, deep analysis, perspectives, and towards performance shifts," *Expert Systems with Applications*, vol. 177, Article ID 114864, 2021.

[31] J. Tu, H. Chen, M. Wang, and A. H. Gandomi, "The colony predation algorithm," *J Bionic Eng*, vol. 18, pp. 674–710, 2021.

[32] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.

[33] L. Sun, S. Chen, J. Xu, and Y. Tian, "Improved monarch butterfly optimization algorithm based on opposition-based learning and random local perturbation," *Complexity*, vol. 2019, Article ID 4182148, 20 pages, 2019.

[34] S. Gupta and K. Deep, "A hybrid self-adaptive sine cosine algorithm with opposition based learning," *Expert Systems with Applications*, vol. 119, pp. 210–230, 2018.

[35] M. J. Aliabadi and M. Radmehr, "Optimization of hybrid renewable energy system in radial distribution networks considering uncertainty using meta-heuristic crow search algorithm," *Applied Soft Computing*, vol. 107, Article ID 10738, 2021.