Improved Horse Herd Optimization with Backtracking Search Algorithm

1st Tanmay Shreshth
Electronics and Electrical Engineering
Indian Institute of Technology Guwahati
Guwahati, India
tanmay.shreshth@iitg.ac.in

2nd Aryan Verma
Electronics and Electrical Engineering
Indian Institute of Technology Guwahati
Guwahati, India
aryanverma@iitg.ac.in

***Index Terms*—Optimization, Evolutionary Algorithm, Swarm Intelligence**

***Abstract*—The Horse herd optimization algorithm (HOA) is a population-based meta-heuristic optimization inspired by horses' herding behaviour. It has a tendency to get trapped in local optima due to loss in diversity along the iterative optimization process. In contrast, the backtracking search optimization algorithm (BSA) has a robust global exploration ability, whereas, it has a low local exploitation ability and converges slowly. This paper proposes an improved HOA with BSA called m-HOA to resolve the original HOA algorithm's problems through BSA's mutation and crossover operators and introduction of the neighborhood. This brings out the strength of both the algorithms: combining local exploitation ability with global exploration ability. The experimental results show that m-HOA outperforms HOA in terms of global exploration ability, accuracy, and rate of convergence on several of the benchmark problems.**

I. Introduction

Optimization is the process of finding the optimum values of a system's state based on some criteria and/or a set of constraints. It is an important area of research with many applications in engineering, sciences and finance. In order to solve an optimization problem, it is required to formulate the real-world problem into a rigorous mathematical form, with the variables that determine the state and the fitness value of the state based on those variables. The mathematical function (called the objective function) could either be required to be minimized or maximized depending upon the scenario [29]. Also, the state variables (referred to as decision variables) need to be within bounds, which describe the search space, and under a given set of constraints, which describe the feasible region. The search space is *n-Dimensional*, where *n* is the number of decision variables. As the size of the search space increases, so does the difficulty in finding a global optimum. Also, the objective function could be linear, non-linear, differentiable, non-differentiable, have many local optima, and so on. All this means that it often becomes quite difficult to determine the optima from classical optimization techniques for real world problems which have huge search spaces, and at times, erratic behaviour [1], [2], [27], [28]. Also, these techniques make use of differential calculus, making them unsuitable in case of non-differentiable or non-continuous objective functions. This leads us to search for alternative methods of which meta-heuristic optimization algorithms are one of the most popular approaches [3]–[5].

Meta-heuristic optimization algorithms are population-based stochastic search algorithms which operate solely based on the fitness value of proposed solutions [30]. They analyse and operate based solely on the fitness value and communicate this information with the outside world (i.e., the user). Although unlike classical techniques they cannot guarantee reaching global optimality, they are easy to use, can be applied to a wide variety of optimization problems, and can be made to perform well even on higher dimensional problems [33]. Also, they can be applied wherever the objective function for an optimization problem is non-linear, non-differentiable and so on. Popular examples include Teaching learning based optimization (TLBO) algorithm [19], Differential evolution (DE) algorithm [18], Particle swarm optimization (PSO) algorithm [20], and genetic algorithms(GA) [21] etc.

Meta-heuristic algorithms need to have two essential capabilities: global exploration ability and local exploitation ability [31], [32]. Global exploration ability means that the algorithm scans the entire search space efficiently, and is able to resist being trapped in locally optimal solutions which are not globally optimum. On the other hand, it should also be able to search for better solutions closer to the solutions already present in population. A fine balance between exploration and exploitation is essential for an algorithm to perform well on a wide variety of problems [34]. Also, most meta-heuristic algorithms contain a set of user-controlled parameters which can be tweaked in order to perform better on different classes of problems, by shifting the balance towards either global exploration or local exploitation.

Evolutionary algorithms (EA) are a sub-class of meta-heuristics based on the principles of evolution first proposed by Charles Darwin (inheritance and natural selection) which occurs in nature. Species evolve in nature and over time, new species emerge, due to various factors and processes.Some of these include mutation of genes, mating of two adults leading to offsprings, crossover of genetic material during mating, and so on [10], [15], [16]. Many of these processes are mimicked using mathematical operators.

Swarm Intelligence (SI) is another sub-class of metaheuristic algorithms that are based on the collective behaviour of social organisms found in nature, such as bees, fish, ant colonies, birds, and so on [8]–[14]. Some popular SI algorithms include Particle Swarm Optimization and Artificial Bee Colony.

(Note that this is not a rigorous classification and is based on the inspiration behind the creation of algorithms. Many EAs frequently depict swarm intelligence, and algorithms can frequently borrow ideas from more than one source).

Horse-herd optimization (HOA) is one such novel optimization algorithm [6], proposed by Naeimi, Azizyan and Rashki (2021). It is inspired by the social behaviour depicted by horses in herds, being specifically targeted towards solving high-dimensional optimization problems, and shows promising results on a few benchmark problems. However, it suffers from poor performance and sub-optimal values on many other benchmark problems, demonstrating premature convergence to local optima. It could be a result of insufficient diversity in its population after many generations (iterations). This hints at poor global exploitation capabilities.

On the other hand, backtracking search optimization (BSA) [7] proposed by Civicioglu in 2013, is another population-based stochastic optimization technique which has a robust global exploration capability but suffers from slow convergence rate. It uses the method of generating trial individuals from current population members. The mutant is generated with the help of mutation and crossover operators, along with the old population. This old population enables BSA to have a memory.

These two algorithms seem complimentary in their strengths so it is proposed to improve the HOA's performance by introducing ideas from BSA. Specifically, mutation and crossover operations, and the concept of memory from previous generations have been borrowed from BSA to improve HOA. Apart from this, we also introduce the concept of neighbourhood, with introducing a concept of neighbourhoods which is unique to our approach [17].

The paper contains brief discussions of the original HOA and BSA after which a detailed description of the modified approach follows. Thereon, experimental results showcasing the performance improvements in contrast to the results obtained from the original algorithm are presented.

II. Horse Herd Optimization

Horse-herd Optimization (HOA) is an optimization algorithm which imitates the social behaviour of horses at different ages using six features, namely grazing, hierarchy, sociability, imitation, defense mechanism and roam [22]–[24].

In HOA, many independent particles are stochastically generated in the search space. Each particle is a potential solution to the optimization problem. Let X_m be the position of the m^{th} horse, and V_m be the velocity of the horse in the current iteration. Then the new position of the horse is given by:

$$X_m^{iter,AGE} = X_m^{iter-1} + V_m^{iter,AGE} \tag{1}$$

Here, V_m models the overall behaviour of the horse, which varies as per its age, and can be broken down into six parts. AGE shows the age range of the considered horse, and *iter* is the current iteration.

Horses exhibit different behaviors at different ages. Maximum lifetime of a horse is about 25–30 years citeb25. In this regard, δ denotes the horses at the age range of 0–5 years, γ indicates the horses at the range of 5–10 years, β shows the horses the age range of 10–15 years, and α demonstrates the horses older than 15 years. A comprehensive matrix of responses should be conducted per iteration to select the age of horses. In this regard, the matrix can be sorted based on the best response and consequently, the first 10 percent of the horses from top of the sorted matrix are selected as α horses. The next 20 percent are in the β group. The γ and δ horses account for 30% and 40% of the remaining horses, respectively. The steps to simulate the six behaviors of the horses are mathematically implemented to detect the velocity vector.

The motion vector of horses at different ages during each cycle of the algorithm can be written as the following, with regard to the above behavior patterns [22], [23], [25]

$$\vec{V}_m^{Iter,\alpha} = \vec{G}_m^{Iter,\alpha} + \vec{D}_m^{Iter,\alpha} \tag{2}$$

$$\vec{V}_m^{Iter,\beta} = \vec{G}_m^{Iter,\beta} + \vec{H}_m^{Iter,\beta} + \vec{S}_m^{Iter,\beta} + \vec{D}_m^{Iter,\beta} \tag{3}$$

$$\vec{V}_m^{Iter,\gamma} = \vec{G}_m^{Iter,\gamma} + \vec{H}_m^{Iter,\gamma} + \vec{S}_m^{Iter,\gamma} + \vec{I}_m^{Iter,\gamma} + \vec{D}_m^{Iter,\gamma} + \vec{R}_m^{Iter,\gamma} \tag{4}$$

$$\vec{V}_m^{Iter,\delta} = \vec{G}_m^{Iter,\delta} + \vec{I}_m^{Iter,\delta} + \vec{R}_m^{Iter,\delta} \tag{5}$$

A. Grazing

Horses are grazing animals, which feed on plants, grasses, forages, etc.

The HOA algorithm models the graze area around each horse with coefficient g as such each horse is grazing on certain areas. Horses graze at all ages throughout their lifetime (α, β, γ and δ horses) [25]. The mathematical implementation of grazing velocity is given by:

$$G_m^{iter,AGE} = g_m^{iter,AGE}(u + P * l)[X_p^{iter-1} - X_m^{iter-1}] \tag{6}$$

$$g_m^{iter,AGE} = g_m^{iter-1,AGE} * \omega_g \tag{7}$$

where, X_p^{iter-1} stands for the best position of that particular horse, and $G_m^{iter,AGE}$ is the motion parameter of i_th horse and it shows the concerned horse's tendency to graze. This factor reduces linearity with ω_g per iteration. l and u are lower and upper bounds of grazing space respectively, and P is a random number between 0 and 1. It is recommended to consider l and u equal to 0.95 and 1.05, respectively, and the coefficient g be equal to 1.5 for all age ranges.

B. Hierarchy

Horses are not free on their own. They pass their lives following a leader, which is often undertaken by human beings. An adult stallion or a mare is also responsible for leadership in the herds of wild horses, which occurs as per the law of hierarchy [26]. In this case, the coefficient *h* in HOA is considered to be the tendency of a herd of horses to follow the most experienced and strongest horse. Studies have shown that the horses follow the law of hierarchy at the middle ages β and γ (aged between 5–15 years). It can be defined as

$$H_m^{iter,AGE} = h_m^{iter,AGE}[X_*^{iter-1} - X_m^{iter-1}] \tag{8}$$

$$h_m^{iter,AGE} = h_m^{iter-1,AGE} * \omega_h \tag{9}$$

where, $H_m^{iter,AGE}$ indicates the effects of the best horse location on the velocity parameter, and X_*^{iter-1} shows the location of the best horse.

C. Sociability

Horses lead a social life and sometimes live with other animals. Herd life increases the horses' security from being hunted by predators. Pluralism increases the chances of survival and makes it easy to escape [25]. This behavior is considered as a movement toward the average position of other horses and is shown by factor s. It is clearly observed that the horses at the ages of 5–15 years (β and γ horses) are interested in the herd, as expressed in the following equation:

$$S_m^{iter,AGE} = s_m^{iter,AGE}\left[\left(\frac{1}{N}\sum_{j=1}^{N} X_j^{iter-1}\right) - X_m^{iter-1}\right] \tag{10}$$

$$s_m^{iter,AGE} = s_m^{iter-1,AGE} * \omega_s \tag{11}$$

D. Imitation

Horses imitate each other, and they learn each other's good and bad habits such as finding the appropriate location of a pasture [26]. The imitation behavior of horses is also considered as the factor *i* in the present algorithm. Young horses try to mimic others, mostly γ horses.

$$I_m^{iter,AGE} = i_m^{iter,AGE}\left[\left(\frac{1}{pN}\sum_{j=1}^{pN} \hat{X}_j^{iter-1}\right) - X_m^{iter-1}\right] \tag{12}$$

$$i_m^{iter,AGE} = i_m^{iter-1,AGE} * \omega_i \tag{13}$$

E_m is the motion vector of i^{th} horse toward the average of best horses with $\hat{X}$ locations. pN shows the number of horses with the best locations. It is proposed that p is set as 10% of the horses. Moreover, ω_i is a reduction factor per cycle for i^{iter} as shown previously.

E. Defense Mechanism

Horses' reaction is a reflection of the fact that they have been victims of predators. They defend themselves by showing the fight-or-flight response. Their first reaction is to escape. Horses fight for food and water to take away the rivals and avoid hazardous environments, where there are enemies such as wolves by instinct [23]. Horses' defense system in the HOA algorithm functions by running away from horses showing inappropriate responses that are far from optimal. Their defense system is characterized by the factor *d*. Horses have to flee or fight against their enemies, as mentioned above. Such a defense mechanism exists throughout the entire lifetime of a young or mature horse, whenever possible (α, β and γ horses). The defense mechanism of horses is presented by a negative coefficient to keep the horse away from inappropriate positions.

$$D_m^{iter,AGE} = -d_m^{iter,AGE}\left[\left(\frac{1}{qN}\sum_{j=1}^{pN} \widetilde{X}_j^{iter-1}\right) - X_m^{iter-1}\right] \tag{14}$$

$$d_m^{iter,AGE} = d_m^{iter-1,AGE} * \omega_d \tag{15}$$

where $D_m^{iter,AGE}$ shows the escape vector of ith horse from the average of some horses with worst locations, which are shown by $\widetilde{X}$ vector. qN is also shows the number of horses with the worst locations. It is suggested that q be set equal to 20 percent of the total horses. ω_d shows the reduction factor per cycle for d^{Iter}, as previously noted.

F. Roam

Horses roam and graze in the nature from pasture to pasture in the search of food. A horse may suddenly go to another location for a graze. Horses are extremely curious, and they often visit everywhere to discover new pastures and to know their neighborhoods [24]. This behavior is simulated as a random movement and shown by a factor r. Roaming in horses is almost exclusively observed at young ages (γ and δ horses) and disappears gradually as they reach maturity.

$$R_m^{iter,AGE} = r_m^{iter,AGE} * P * X_m^{iter-1} \tag{16}$$

$$r_m^{iter,AGE} = r_m^{iter-1,AGE} * \omega_r \tag{17}$$

where, $R_m^{iter,AGE}$ represents the random velocity vector of i^{th} horse for a local search and an escape from local minima, and ω_r shows the reduction factor of $r_m^{iter,AGE}$

The general velocity vector for each horse is summarised below:

Velocity for δ horses (aged 0-5 years):

$$V_m^{iter,\delta} = \Big[r_m^{iter-1,\delta} * \omega_r * P * X_m^{iter-1}\Big] + \Big[i_m^{iter-1,\delta} * \omega_i \Big[\Big(\frac{1}{pN}\sum_{j=1}^{pN} \hat{X}_j^{iter-1}\Big) - X_m^{iter-1}\Big]\Big] + \Big[g_m^{iter-1,\delta} * \omega_g (u + P * l)[X_p^{iter-1} - X_m^{iter-1}]\Big] \tag{18}$$

Velocity for γ horses (aged 5-10 years):

$$V_m^{iter,\gamma} = \Big[s_m^{iter-1,\gamma} * \omega_s \Big[\Big(\frac{1}{N}\sum_{j=1}^{N} X_j^{iter-1}\Big) - X_m^{iter-1}\Big]\Big] + \Big[i_m^{iter-1,\gamma} * \omega_i \Big[\Big(\frac{1}{pN}\sum_{j=1}^{pN} \hat{X}_j^{iter-1}\Big) - X_m^{iter-1}\Big]\Big] + \Big[g_m^{iter-1,\gamma} * \omega_g (u + P * l)[X_p^{iter-1} - X_m^{iter-1}]\Big] + \Big[h_m^{iter-1,\gamma} * \omega_h [X_*^{iter-1} - X_m^{iter-1}]\Big] + \Big[r_m^{iter-1,\gamma} * \omega_r * P * X_m^{iter-1}\Big] - \Big[d_m^{iter-1,\gamma} * \omega_d \Big[\Big(\frac{1}{qN}\sum_{j=1}^{pN} \widetilde{X}_j^{iter-1}\Big) - X_m^{iter-1}\Big]\Big] \tag{19}$$

Velocity for β horses (aged 10-15 years):

$$V_m^{iter,\beta} = \Big[s_m^{iter-1,\beta} * \omega_s \Big[\Big(\frac{1}{N}\sum_{j=1}^{N} X_j^{iter-1}\Big) - X_m^{iter-1}\Big]\Big] + \Big[g_m^{iter-1,\beta} * \omega_g (u + P * l)[X_p^{iter-1} - X_m^{iter-1}]\Big] + \Big[h_m^{iter-1,\beta} * \omega_h [X_*^{iter-1} - X_m^{iter-1}]\Big] - \Big[d_m^{iter-1,\beta} * \omega_d \Big[\Big(\frac{1}{qN}\sum_{j=1}^{pN} \widetilde{X}_j^{iter-1}\Big) - X_m^{iter-1}\Big]\Big] \tag{20}$$

Velocity for α horses (older than 15 years):

$$V_m^{iter,\alpha} = \Big[g_m^{iter-1,\alpha} * \omega_g (u + P * l)[X_p^{iter-1} - X_m^{iter-1}]\Big] - \Big[d_m^{iter-1,\alpha} * \omega_d \Big[\Big(\frac{1}{qN}\sum_{j=1}^{pN} \widetilde{X}_j^{iter-1}\Big) - X_m^{iter-1}\Big]\Big] \tag{21}$$

III. Backtracking Search Algorithm

Backtracking search algorithm (BSA) is an evolutionary algorithm which employs a unique mechanism for generating a trial individual for fast exploration, i.e. it has three genetic operators namely selection, mutation and crossover. In the mutation strategy employed, there is only one direction individual for each target individual,which is quite different from the conventional one used in differential evolution (DE) and its variants. It also involves a non-uniform crossover strategy and uses memory in which it stores a population from a randomly chosen previous generation for use in generating the search-direction matrix.

BSA can be explained by dividing its functions into five processes as is done in other EAs: initialization, selection-I, mutation, crossover and selection-II.

A. Initialization

BSA initializes the population P with 22

$$P_{i,j} \sim U(low_j, up_j) \tag{22}$$

for i = 1,2,3,...,N and j = 1,2,3,...,D, where N and D are the population size and the problem dimension, respectively, U is the uniform distribution and each P_i is a target individual in the population P.

B. Selection- I

BSA's Selection-I stage determines the historical population *oldP* to be used for calculating the search direction. The initial historical population is determined using:

$$oldP_{i,j} \sim U(low_j, up_j) \tag{23}$$

BSA has the option of redefining oldP at the beginning of each iteration through the 'if-then' rule in Eq. 24:

$$if \;\; a < b \;\; then \;\; oldP := P | a, b \sim U(0,1) \tag{24}$$

where := is the update operation. This equation ensures that BSA designates a population belonging to a randomly selected previous generation as the historical population and remembers this historical population until it is changed. Thus, BSA has a memory.

After oldP is determined, Eq. 25 is used to randomly change the order of the individuals in oldP:

$$oldP := permuting(oldP) \tag{25}$$

The permuting function used in Eq. 25 is a random shuffling function.

C. Mutation

BSA's mutation process generates the initial form of the trial population Mutant using Eq. 26.

$$Mutant = P + F(oldP - P) \tag{26}$$

In Eq. 26, F controls the amplitude of the search-direction matrix (oldP - P). Because the historical population is used in the calculation of the search-direction matrix, BSA generates a trial population, taking partial advantage of its experiences from previous generations. F can be chosen from a number of options including standard walk and Brownian walk, based on what works best in a given scenario.

D. Crossover

BSA's crossover process generates the final form of the trial population T. The initial value of the trial population is Mutant, as set in the mutation process. Trial individuals with better fitness values for the optimization problem are used to evolve the target population individuals. BSA's crossover process has two steps. The first step calculates a binary integer-valued matrix (map) of size N_D that indicates the individuals of T to be manipulated by using the relevant individuals of P. If $map_{n,m} = 1$, where n $\in$ 1, 2, 3, ... , N and m$\in$ 1, 2, 3, ..., D, T is updated with $T_{n,m} := P_{n,m}$

BSA's crossover strategy is entirely different from the crossover strategies used in DE and its variants. The mix rate parameter (mixrate) in the BSA's crossover process controls the numbers of elements of individuals that will mutate in a trial using mixrate. The function of the mix rate is entirely different from the crossover rate used in DE. To determine the matrix map, two predefined strategies are randomly used. The first strategy uses parameter mixrate. The second strategy allows the random selection of only one individual to change each trial individually. Some individuals of the T, obtained at the end of the BSA's crossover process, can overflow the allowed search space limits due to the BSA's mutation strategy. The individuals beyond the search space limits should be regenerated.

E. Selection -II

In the selection-II, if T_i has better fitness value than the corresponding P_i, it is used to update P_i based on greedy selection. If the best individual of the population has a better fitness value than the BSA's optimum global value, the optimum global value will be updated as well.

IV. Modified Horse-herd Optimization

This section contains the modified approach to horse herd optimization by combining the original Horse-herd optimization and the backtracking search optimization algorithm.

A. Motivation

The performance of original HOA shows considerable scope for improvement. It displays premature convergence to local optima on quite a few benchmark problems. This could possibly be explained by limited global exploration ability and lack of diversity in population as the iterations go on.

One of the most crucial aspects of any population-based meta-heuristic optimization algorithm is balancing the extents of local exploitation (for rapid convergence) and global exploration (to avoid local optima traps).

On the other hand, Backtracking search optimization algorithm demonstrates the ability to increase the diversity of particles, resulting in increased global exploitation. Two crucial operations that help it to achieve this are mutation and crossover. In addition to this, as mentioned before, BSA also has a memory in the form of old population matrix, which means that it takes advantage of experiences from its past generations when it generates a mutant in the current generation.

Apart from this, we also introduce another concept- the concept of neighbourhoods. The whole herd is divided into neighbourhoods, and the best position of the neighbourhood as a whole, is used to guide the horses of that particular neighbourhood instead of their personal best positions.

Utilizing all of this, we propose a modified Horse-herd optimization which improves the original HOA's performance on a number of benchmark problems.

B. Mutation and Crossover

After calculating the initial form of the trial population member T_m using Eq.1, mutation and crossover operations are performed to generate the offsprings. As such, there are slight modifications in Eq.26. Let O_m refer to the offspring of the m^{th} population member. Then, as per Eq.26 and the subsequent crossover operation:

$$O_m = T_m + (map * F) * (X_m^{old} - T_m) \tag{27}$$

Here, map is the binary array as described in section III-D, and X_m^{old} is the m^{th} population member of the old population (described by oldP in BSA).

C. Neighbourhood

In general, horses also roam about in smaller groups within a larger herd [25]. This behaviour can be mimicked in this algorithm by introducing the concept of neighbourhoods, which resemble these small groups. So, a horse would tend to follow the strongest horse in its neighbourhood. This concept has been introduced to increase the convergence rate which would be affected by introduction of mutation and crossover.

Here, Eq.6 gets modified; from using the personal best performance of each horse to utilising the best performance of the whole neighbourhood of horses, represented by X_L^{iter-1}.

$$G_m^{iter,AGE} = g_m^{iter,AGE}(u + P * l)[X_L^{iter-1} - X_m^{iter-1}] \tag{28}$$

This change leads to a corresponding change in the velocity equations Eqs. 18, 19, 20, and 21 for all horses of all ages.

D. Implementation Procedure

1) **Input:** objective function (OF), number of decision variables (nVar), population size (nHorse), neighbourhood size (ns), and number of iterations to be run (MaxIt).
2) **Output:** the global optimum value found at the end of all iterations (Zopt) and the corresponding solution in the population (ZMopt).
3) **Initialization:** the population matrix, old population matrix, and evaluation of the fitness values of their

members. Also, randomly assign a neighbourhood to each horse, making sure that all neighbourhoods contain equal no. of members. Also, initialize the personal bests, local bests (best of each neighbourhood) and the global best solution arrays and update them accordingly.

4) **Parameters:** Set the parameters used in Eqs. 6-17, and the mixrate.
5) **Iterations:** The iterations would run through until they reach the stopping criterion (MaxIt):
   5.1 **Sort the horses** by their fitness value, and subsequently their age. Determine the average (Eq.10), good (Eq.12), and bad (Eq.14) positions. Now loop through each member of the population, i.e., individual horses:
      5.1.1 Determine the horse's age, calculate its velocity (Eqs.18-21) accordingly, and bound it.
      5.1.2 Add the velocity to the current position, resulting in the **trial position**.
      5.1.3 In order to perform mutation, the operands **F** and **map** are generated.
      5.1.4 Generate the offspring as per Eq.27 and bound it within the domain.
      5.1.5 Update the old population matrix as the current population *if* $a < b$, where a & b are two random numbers taken from a uniform distribution between 0 and 1. Also, randomly shuffle the old population matrix.
      5.1.6 Calculate the fitness of the offspring and update the population if the offspring's fitness is better than its parent.
      5.1.7 Similarly, update the personal, local and global bests based on greedy selection.
   5.2 Successfully loop through all population members and update the value of parameters affected by the damping ratio (ω).
6) Stop when the stopping criteria is met (such as after specified iterations).

V. Results

This section contains the details of the experimental tests, analysis and optimization used in the proposed Modified Horse Herd Optimization (m-HOA) along with comparison to the original Horse Herd Optimization (HOA) on several well known benchmark functions. All the tests have been performed on Matlab Online as the software base. The benchmark functions used here are quite prevalent in literature [35]–[38]. These all are minimization problems, being a mix of fixed dimensio n unimodal functions, multi dimensional multimodal functions and fixed dimensional multimodal problems. A special focus on multimodal problems has been done to test the global search ability of m-HOA, which was seen to be poor in HOA.

The stochastic nature of metaheuristic algorithms makes it difficult to accurately compare different results, therefore statistical measures such mean and standard deviation have employed to provide a sound comparison over a sample of results.

The following parameter values were used for the experiments:

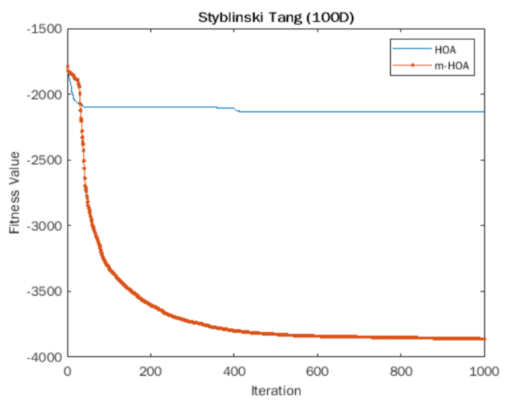
- for HOA:
  - number of iterations: 1000
  - population size: 50
  - Damping ratio ω (all): 1.0
  - q = 0.02
  - p = 0.02
  - g^{α} = 1.50
  - d^{α} = 0.5
  - h^{α} = 1.5
  - g^{β} = 1.50
  - h^{β} = 0.9
  - s^{β} = 0.20
  - d^{β} = 0.20
  - g^{γ} = 1.50
  - h^{γ} = 0.50
  - s^{γ} = 0.10
  - i^{γ} = 0.30
  - d^{γ} = 0.10
  - r^{γ} = 0.05
  - g^{δ} = 1.50
  - r^{δ} = 0.10
- for m-HOA (unmentioned parameters were identical to that above):
  - number of iterations: 1000
  - population size: 50
  - neighbourhood size = 5
  - Damping ratio ω (all): 0.9
  - mixrate: 1.0
  - F: 4 * randg

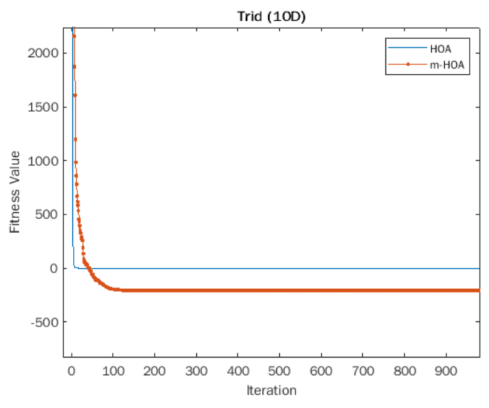
For each benchmark function, the experiment was done 10 times to have a substantial sample on which base inferences on. It can be seen from the table below, the m-HOA proposed here leads to much better optimal values in essentially the same number of functional evaluations.

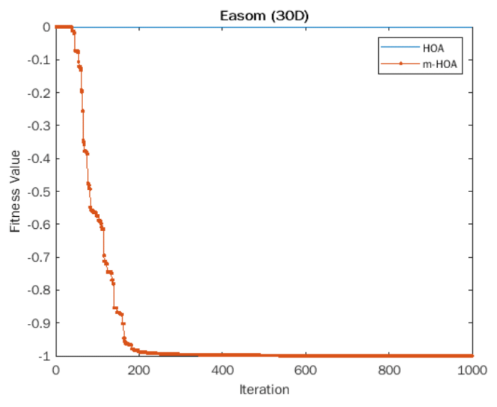
Further plots comparing convergence rates of original and proposed modified approach have been provided to strengthen our inferences.

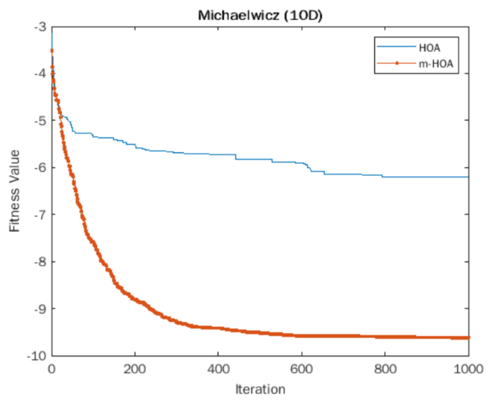
Functions	Equation	Range	Optimum	HOA	m-HOA
Colville (4D)	$100(x_1^2-x_2)^2+(x_1-1)^2+(x_3-1)^2+90(x_3^2-x_4)^2$ $+10.1((x_2-1)^2+(x_4-1)^2)+19.8(x_2-1)(x_4-1)$	[-10,10]	0		
mean				3.60286818	2.41188E-08
Std Dev.				1.879463881	3.95303E-08
Best				1.13360479	9.87501E-13
Easom (10D)	$(-1)^d\Big(\prod_{i=1}^{d}\cos^2(x_i)\Big)exp\Big[-\sum_{i=1}^{n}(x_i-\pi)^2\Big]$	$[-2\pi,2\pi]$	-1		
Mean				-0.806581902	-0.999999032
Std Dev.				0.222460132	3.06154E-06
Best				-0.992035093	-1
Easom (30D)	$(-1)^d\Big(\prod_{i=1}^{d}\cos^2(x_i)\Big)exp\Big[-\sum_{i=1}^{n}(x_i-\pi)^2\Big]$	$[-2\pi,2\pi]$	-1		
Mean				0	-0.999980367
Std Dev.				0	3.92819E-05
Best				0	-1
Michalewicz (5D)	$-\sum_{i=1}^{d}sin(x_i)sin^{2m}(\frac{ix_i^2}{\pi})$	$[0,\pi]$	-4.6876858		
Mean				-4.286189212	-4.687658179
Std Dev.				0.333750196	5.88759E-13
Best				-4.60049743	-4.687658179
Michalewicz (10D)	$-\sum_{i=1}^{d}sin(x_i)sin^{2m}(\frac{ix_i^2}{\pi})$	$[0,\pi]$	-9.66015		
Mean				-6.269095049	-9.599316075
Std Dev.				0.801561365	0.064423606
Best				-7.32810833	-9.660151716
Modified Langermann (10D)	$-\sum_{i=1}^{m}\Big(c_j cos(d_j/\pi)exp(-\pi d_j)\Big), d_j=\sum_{j=1}^{d}(x_j-A_{ij})^2$	$[0,10]$	-0.965		
Mean				-0.307374374	-0.57070765
Std Dev.				0.288497726	0.262807503
Best				-0.7053177550	-0.705525179
Perm Function (D, beta) (2D)	$\sum_{i=1}^{d}\Big(\sum_{j=1}^{d}(j_i+\beta)\Big(\Big(\frac{x_j}{j}\Big)^i-1\Big)\Big)^2$	[-2,2]	0		
Mean				1.76073E-06	0
Std Dev.				3.0663E-06	0
Best				3.30396E-10	0
Perm Function (D, beta) (5D)	$\sum_{i=1}^{d}\Big(\sum_{j=1}^{d}(j_i+\beta)\Big(\Big(\frac{x_j}{j}\Big)^i-1\Big)\Big)^2$	[-5,5]	0		
Mean				317.1697127	0.109338151
Std Dev.				400.3182494	0.15936805
Best				31.59208796	0.0041797
Perm Function (0, D, beta) (5D)	$\sum_{i=1}^{d}\Big(\sum_{j=1}^{d}(j+\beta)(x_j^i-\frac{1}{j^i})\Big)^2$	[-5,5]	0		
Mean				6.924621745	0.025707552
Std Dev.				3.13278806	0.046759022
Best				2.111677761	3.15448E-07

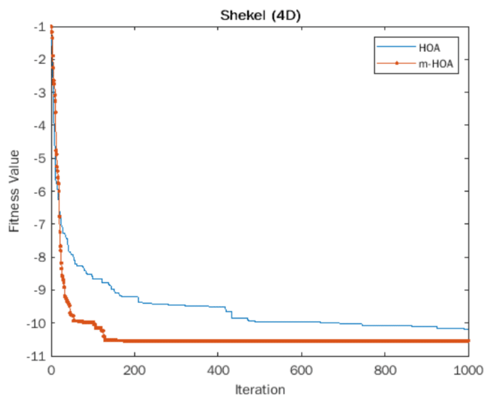
Functions	Equation	Range	Optimum	HOA	m-HOA
Power Sum (4D)	$\sum_{i=1}^{d}\Big[\Big(\sum_{j=1}^{d} x_j^i\Big) - b_i\Big]^2$	[0,4]	0		
Mean				0.239616435	0.002328661
Std Dev.				0.249090057	0.003793212
Best				0.072680439	5.87414E-06
Rosenbrock (5D)	$\sum_{i=1}^{d-1}\Big[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\Big]$	[-5,10]	0		
Mean				3.33354967	0.000994324
Std Dev.				0.957161174	0.001327506
Best				1.48114315	5.05419E-07
Rosenbrock 2.0 (30D)	$\sum_{i=1}^{d-1}\Big[100(x_i+1-x_i^2)^2+(x_i-1)^2\Big]$	[-5,5]	0		
Mean				1501.152528	43.41484831
Std Dev.				98.40209302	8.94276239
Best				1327.530228	28.90789221
Schwefel (5D)	$418.9829d-\sum_{i=1}^{d} x_i sin(\sqrt{\lvert x_i\rvert})$	[-500,500]	0		
Mean				166.1823141	0.000358547
Std Dev.				119.9843864	0.000931914
Best				38.75635797	6.36378E-05
Shekel (4D)	$-\sum_{i=1}^{m}\Big(\sum_{j=1}^{4}(x_j-C_{ji})^2+\beta_i\Big)^{-1}$	$[0,10]$	-10.5364		
Mean				-9.684084761	-9.243129822
Std Dev.				1.799734073	2.726543903
Best				-10.4727897	-10.53644315
Styblinski Tang (10D)	$\frac{1}{2}\sum_{i=1}^{d}(x_i^4-16x_i^2+5x_i)$	$[-5,5]$	-391.6599		
Mean				-371.5254763	-391.661657
Std Dev.				23.46239407	0
Best				-388.1531958	-391.661657
Styblinski Tang (100D))	$\frac{1}{2}\sum_{i=1}^{d}(x_i^4-16x_i^2+5x_i)$	$[-5,5]$	-3916.599		
Mean				-2111.188834	-3907.598602
Std Dev.				82.16151967	7.129719273
Best				-2255.343651	-3915.910163
Trid (10D)	$\sum_{i=1}^{d}(x_i-1)^2-\sum_{i=2}^{d}(x_ix_{i-1})$	$[-100,100]$	-210		
Mean				-1.154766703	-209.99999448
Std Dev.				12.80878943	1.20644E-05
Best				-34.56815121	-210
Trid (20D)	$\sum_{i=1}^{d}(x_i-1)^2-\sum_{i=2}^{d}(x_ix_{i-1})$	$[-400,400]$	-1520		
Mean				11.81613462	-1500.935659
Std Dev.				4.990207401	15.84016939
Best				-0.79409643	-1517.220452

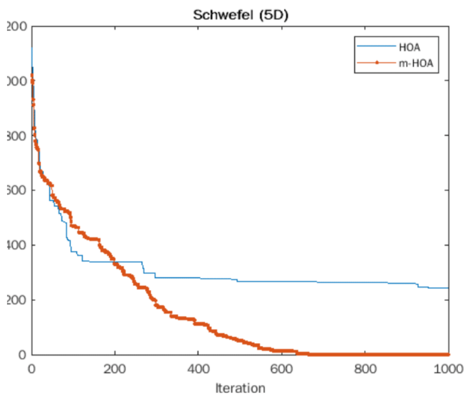

Styblinski Tang (100D)
HOA
m-HOA
Fitness Value
Iteration
-1500
-2000
-2500
-3000
-3500
-4000
0
200
400
600
800
1000


Trid (10D)
HOA
m-HOA
Fitness Value
Iteration
2000
1500
1000
500
0
-500
0
100
200
300
400
500
600
700
800
900


Easom (30D)
HOA
m-HOA
Fitness Value
Iteration
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
-0.9
-1
0
200
400
600
800
1000


Michaelwicz (10D)
HOA
m-HOA
Fitness Value
Iteration
-3
-4
-5
-6
-7
-8
-9
-10
0
200
400
600
800
1000


Shekel (4D)
HOA
m-HOA
Fitness Value
Iteration
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
0
200
400
600
800
1000


Schwefel (5D)
HOA
m-HOA
Iteration
200
0
0
200
400
600
800
1000

VI. Conclusion

This paper proposed a modified Horse Herd Optimization Algorithm employing ideas such as mutation and crossover operations, and memory, inherited from Backtracking search algorithm, to improve upon the poor global exploration capabilities displayed by Horse Herd Optimization resulting in premature convergence and getting trapped in local optima on some benchmark problems. Also, we introduced the idea of neighbourhoods in order to offset the slow convergence rate brought upon by these evolutionary operations. The results and plots show that the proposed algorithm successfully outperforms the original algorithm on benchmark functions. Further work can include improvements towards time complexity of the modified approach and extensive analysis of the parameter sensitivity towards reaching the optima.

References

[1] Gharehchopogh FS, Shayanfar H, Ghoglizadeh H (2019) A comprehensive survey on symbiotic organisms search algorithms. Artif Intell Rev 53:1–48

[2] Stodola P (2020) Hybrid ant colony optimization algorithm applied to the multi-depot vehicle routing problem. Nat Comput 19:1–13

[3] Gharehchopogh FS, Gholizadeh H (2019) A comprehensive survey: Whale Optimization Algorithm and its applications. Swarm Evol Comput 48:1–24

[4] Srivastava S, Sahana SK (2019) A survey on traffic optimization problem using biologically inspired techniques. Nat Comput 19:1–15

[5] Benyamin A, Farhad SG, Saeid B (2021) Discrete farmland fertility optimization algorithm with metropolis acceptance criterion for traveling salesman problems. Int J Intell Syst 36(3):1270–1303

[6] Farid MiarNaeimi and Gholamreza Azizyan and Mohsen Rashki (2021) Horse herd optimization algorithm: A nature-inspired algorithm for high-dimensional optimization problems, Knowledge-Based Systems vol 213, https://doi.org/10.1016/j.knosys.2020.106711

[7] Pinar Civicioglu (2013) Backtracking Search Optimization Algorithm for numerical optimization problems, Applied Mathematics and Computation 219 (2013) 8121–8144

[8] K. Deb, A. Pratap, S. Agarwal, A fast and elitist multiobjective genetic algorithm: Nsga-II, IEEE Trans. Evol. Comput. 6 (2002) 182–197.

[9] M.G. de Carvalho, A.H.F. Laender, M.A. Goncalves, et al, A genetic programming approach to record deduplication, IEEE Trans. Knowl. Data. Eng. 24 (2012) 399–412.

[10] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2009) 398–417

[11] H. Ishibuch, T. Yoshida, T. Murata, Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling, IEEE Trans. Evol. Comput. 7 (2003) 204–223.

[12] J.K. Kishore, L.M. Patnaik, V. Mani, et al, Application of genetic programming for multicategory pattern classification, IEEE Trans. Evol. Comput. 4 (2000) 242–258.

[13] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (2006) 646–657.

[14] M.F. Tasgetiren, P.N. Suganthan, Q.K. Pan, An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem, Appl. Math. Comput. 215 (2010) 3356–3368.

[15]] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (2009) 945–958.

[16] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif. Intell. Rev. 33 (2010) 61–106.

[17] Hamid Reza Rafat Zaman and Farhad Soleimanian Gharehchopogh, An improved particle swarm optimization with backtracking search (2021), Engineering with Computers, optimization algorithm for solving continuous optimization problems

[18] Storn R, Price K, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces(1997). J Glob Optim 11(4):341–359

[19] Rao RV, Savsani VJ, Vakharia D, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems (2011). Comput Aided Des 43(3):303–315

[20] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.

[21] Bäck, T., Fogel, D. B. and Michalewicz, Z., 1997, Handbook of Evolutionary Computation, Oxford University Press, Oxford.

[22]] G.H. Waring, Horse behavior, in: The Behavioral Traits and Adaptations of Domestic and Wild Horses, Including Ponies, Noyes Publications, Mill Road, 1983.

[23] S.M. McDonnell, The Equid Ethogram: A Practical Field Guide to Horse Behavior, Eclipse Press, 2003.

[24] M.A. Levine, Domestication and early history of the horse, in: The Domestic Horse: The Origins, Development and Management of Its Behaviour, 2005, pp. 5–22.

[25] K. Krueger, J. Heinze, Horse sense: Social status of horses (Equus Caballus) affects their likelihood of copying other horses' behavior, Anim. Cogn. 11 (3) (2008) 431–439

[26] F. Bogner, A comprehensive summary of the scientific literature on horse assisted education in Germany, 2011

[27] Rao, S.S. (2019). Classical Optimization Techniques. In Engineering Optimization Theory and Practice, S.S. Rao (Ed.). https://doi.org/10.1002/9781119454816.ch2

[28] Tsai, H.-R and Chen, Toly. (2014). Enhancing the Sustainability of a Location-Aware Service through Optimization. Sustainability (Switzerland). 6. 9441-9455. 10.3390/su6129441.

[29] Stephen Boyd and Lieven Vandenberghe. 2004. Convex Optimization. Cambridge University Press, USA.

[30] Hussain, Kashif and Salleh, Mohd and Cheng, Shi and Shi, Yuhui. (2019). Metaheuristic research: a comprehensive survey. Artificial Intelligence Review. 52. 10.1007/s10462-017-9605-z.

[31] Akkuş, Özge and Demir, Emre. (2016). COMPARISON OF SOME CLASSICAL AND META-HEURISTIC OPTIMIZATION TECHNIQUES IN THE ESTIMATION OF THE LOGIT MODEL PARAMETERS.. International Journal of Advanced Research. 4. 1026-1042. 10.21474/IJAR01/1890.

[32] Beheshti, Zahra and Shamsuddin, Siti Mariyam. (2013). A review of population-based meta-heuristic algorithm. International Journal of Advances in Soft Computing and Its Applications. 5. 1-35.

[33] Dankolo, Nasiru and Radzi, Nor and Sallehuddin, Roselina and Mustaffa, Noorfa. (2017). A study of metaheuristic algorithms for high dimensional feature selection on microarray data. AIP Conference Proceedings. 1905. 040010. 10.1063/1.5012198.

[34] Sá, Angela and Andrade, Adriano and Soares, Alcimar and Nasuto, Slawomir. (2008). Exploration vs. Exploitation in Differential Evolution. Biomedical Engineering. 1-7.

[35] Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. Retrieved December 3, 2021, from http://www.sfu.ca/ ssurjano.

[36] Global Optimization Test Problems. Retrieved June 2013, from http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm.

[37] Yang XS (2010) Test problems in optimization. arXiv: 1008. 0549

[38] Ali, M.M., Khompatraporn, C. Zabinsky, Z.B. A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems. J Glob Optim 31, 635–672 (2005). https://doi.org/10.1007/s10898-004-9972-2