Received 21 April 2024, accepted 11 June 2024, date of publication 26 June 2024, date of current version 15 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3419172

RESEARCH ARTICLE

Dholes Hunting–A Multi-Local Search Algorithm Using Gradient Approximation and Its Application for Blockchain Consensus Problem

BINH MINH NGUYEN, THIEU NGUYEN, QUOC-HIEN VU, TRAN HUY HUNG, TRAN HOANG HAI, HUYNH THI THANH BINH, AND VAN-DANG TRAN
School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi 100000, Vietnam

Corresponding author: Binh Minh Nguyen (minhnb@soict.hust.edu.vn)

This work was supported in part by Vietnam Ministry of Education and Training grant entitled "Research and Development of Network Intrusion Detection Framework Based on Combination and Improvement of Machine Learning and Deep Learning Techniques" under Grant BKA-04-2022, and in part by the Center for Digital Technology and Economy (BK Fintech) of Hanoi University of Science and Technology (HUST).

ABSTRACT This study introduces a brand-new swarm-inspired algorithm dubbed dholes hunting-based optimization (DhoH) based on an animal hunting strategy to solve global optimization problems. The technique is a brilliant idea for simultaneously finding many local minima. The dhole's hunting strategy is coordinated by members of a swarm, clustering and chasing prey. A clustering approach and finding an optimal global algorithm describe primarily based on gradient approximation. We use four benchmark function datasets to evaluate the DhoH's performance. We compare the achieved results with several previous research from various well-known algorithms. The comparisons demonstrate that DhoH is better than other meta-heuristic algorithms in most cases and determines high-quality solutions with fewer control parameters. Besides, we also explore the application of DhoH in optimizing the decentralized level of Meta-heuristic Proof of Criteria consensus protocol (MPoC) in Blockchain Network to further demonstrate its potential in multi-dimensional problems. The results show the superior effectiveness of DhoH in terms of computational burden and solution precision compared with the existing optimization techniques in the literature.

INDEX TERMS Dholes hunting-based optimization, nature-inspired computing, swarm-inspired meta-heuristics, global search optimization, MPoC consensus protocol, blockchain network.

I. INTRODUCTION

Natural meta-heuristic algorithms are usually inspired by social communication behaviors and the food-seeking of creatures in nature. Many algorithms have proven highly applicable when solving many NP-complete problems with optimal or near-optimal results. For problems with many local optimal points, employing deterministic algorithms like fast steepest, sequential quadratic programming, conjugate gradient, and quasi-Newton methods [1], [2], [3], [4] is commonly very difficult. Meta-heuristic algorithms are, on the other hand, a non-deterministic method that uses different strategies and operators iteratively to explore and exploit the search space supported by a minimum or maximum function [5]. Therefore, meta-heuristics have been widely employed in science and real-life applications like economy and finance, scheduling, and engineering design applications [6], [7], [8], [9].

Swarm-inspired techniques are population-based meta-heuristics replicating wild creatures' social interaction and collective behavior like birds, cats, and microbes [10]. Swarm-based algorithms primarily update and generate new solutions based on the global best solution and their neighbors. These algorithms underscore the ability to explore

The associate editor coordinating the review of this manuscript and approving it for publication was Thanh Ngoc Dinh.

based on the diversity of platforms and develop exploitation based on searching for the best solution. Despite the appearance-after evolution algorithms [11], the development of this type of algorithm is the fastest and most powerful. We listed the well-known algorithms from 1970 onwards in Table 1.

The No Free Lunch Theorem of Optimization [53] states that while various swarm-based techniques have been created, no algorithm can effectively address every type of optimization issue. Furthermore, the bulk of current techniques still leaves out several vital difficulties. *First*, it is challenging to reconcile the two typical characteristics of meta-heuristics inspired by nature: exploitation and exploration. Exploitation is the process of probing a local area to identify a promising answer, whereas exploration is leaving any particular region and exploring uncharted territory. Exploration should typically occur early in the development of algorithms, while exploitation happens later [54]. When looking for the global optimum for a particular problem, exploration and exploitation must coexist harmoniously.

Second, to compete with the existing state-of-the-art optimization algorithms, it is required to provide an algorithm with few control parameters because many methods have several control parameters, making it challenging to select the appropriate parameter set for diverse tasks. As seen in Table 1, the listed studies above show that behaviors inspired by a predator from nature can be efficiently simulated under the mathematical form to solve an optimization problem. In this study, we also take inspiration from the hunting behaviors of dholes as a group and a single individual as a local search, reproducing, and moving to a new pack process as the global search to design a new algorithm called Dholes Hunting-based Optimization (DhoH). This method not only combines the power of both exploration and exploitation but also requires very few control parameters.

The proposed DhoH algorithm may be used in several real-world situations, much as previous meta-heuristic techniques. We demonstrate its effectiveness in improving a blockchain consensus mechanism in real-world experiments. Blockchain, a distributed ledger with visible and impenetrable transactions, is a new technology that may raise network security measures [55]. Consensus techniques ensure that blockchain systems operate steadily and assist all network nodes in transaction verification. We use DhoH to maximize the decentralized level of Meta-heuristic Proof of Criteria (MPoC), a successful protocol recently presented in [56], to demonstrate its capabilities.

To be more precise, we listed our contribution as follows:

- Based on the movement and hunting behavior of dholes, we suggest an effective technique for global optimum search, which involves grouping dholes for prey-seeking, utilizing cooperative techniques among dholes in groups, and employing strategies for the movement and pursuit of dholes and prey. The algorithm employs a K-means clustering approach and cluster optimization, utilizing gradient approximation. The combined strategies result in robust performance when the model operates in complex domains, quickly achieving global/local optimal points.
- The algorithm exhibits fast and stable convergence rates and can readily be applied to various problems. Thirty benchmark functions, including uni, multi, hybrid, and composition modal, were used to measure the algorithm's performance. The outcomes demonstrate the exceptional benchmark performance and efficiency of DhoH, particularly in complicated spaces and domains.
- The model is implemented to evaluate its effectiveness on a blockchain network model and has outperformed other state-of-the-art experimental algorithms.

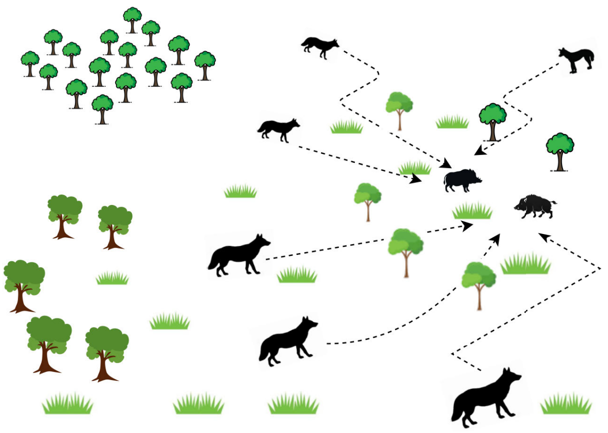


FIGURE 1. Visualization of natural hunting behavior of dholes.

The rest of this paper is designed in the following details: Section II presents our inspiration and mathematical form of the proposed DhoH algorithm. The theoretical experiment and practical experiment are shown in Section III. Section IV discusses the results of the proposed DhoH and several state-of-the-art algorithms in both the global searching test and application test. Our conclusion and some recommendations for future works are shown in section V.

II. DHOLES HUNTING-BASED OPTIMIZATION (Dhoh)

A. INSPIRATION

The Dhole (Cuon alpinus) is called the Red Dog, Asiatic Wild Dog, and Whistling Dog. It is a member of one of the five canid species found throughout the Tibetan Plateau. The wolf (Canis lupus), three fox species (red fox (Vulpes vulpes), Tibetan fox (V. ferrilata), and sand fox) round out the group (V. corsac)u [57]. Dholes live in a broad range of environments, including alpine regions, scrubby areas, thickly forested areas, and steppes.[1]

Dholes are gregarious creatures that live in packs. The pack works together to eat, meet its requirements, and care for itself. Each pack has three to twelve members, and the packs occasionally collaborate or have fun with dholes from other packs. Dhole packs will occasionally band together to create a super pack of up to 30 animals. In Fig. 1, dholes

[1]https://animals.sandiegozoo.org/animals/dhole

TABLE 1. Review of swarm-inspired meta-heuristic algorithms since the 70s.

Year of proposal	Algorithm	Source of inspiration
1995	Particle Swarm Optimization [12]	Bird flock
1999	Ant colony optimization [13]	Ant Colony
2001	Honey Bees Optimization Algorithm [14]	Marriage in honey-bees
2002	Bacterial Foraging Optimization [15]	Bacteria herds
2003	Fish Swarm Algorithm [16]	Fish swarm
2005	Artificial Bee Colony [17]	Honey bee swarm
2006	Cat Swarm Optimization [18]	Cat behaviors
	Termite Algorithm [19]	Termite colony
2007	Wolf pack search [20]	Wolf herd
	Monkey Search [21]	Monkey swarm
2008	Bee Collecting Pollen Algorithm [22]	Bees
2009	Cuckoo Search [23]	Cuckoo herd
	Dolphin Partner Optimization [24]	Dolphin Swarm
2010	Bat Algorithm [25]	Bat herd
	Firefly Algorithm [26]	Firefly
2011	Bumble bees mating optimization [27]	Bumble bees swarm
2012	Fruit-fly Optimization Algorithm [28]	Fruit-fly
	Krill Herd Optimization [29]	Krill Herd
2013	Bird Mating Optimizer [30]	Bird Mating
	Dolphin Echolocation [31]	Dolphin
2014	Chicken Swarm Optimization [32]	Chicken Swarm
	Wolf Pack Algorithm [33]	Wolf Pack hunting behavior
	Grey Wolf Optimizer [34]	Grey Wolf hunting behavior
	Spider Monkey Optimization [35]	Spider Monkey
2015	Moth-flame optimization [36]	Orientation of Moth-flame
	Monarch Butterfly optimization [37]	Migration of Monarch Butterfly
2016	Shark smell optimization [38]	Shark smell
	Crow Search Algorithm [39]	Crow herd
	Whale optimization [40]	Hunting behavior of whale herd
	Bird Swarm Algorithm [41]	Social interactions in bird swarm
	Lion Optimization Algorithm [42]	Cooperation characteristics of Lion
2017	Grasshopper Optimization [43]	Grasshoppers searching for food
	Salp Swarm Algorithm [44]	Salp Swarm
	Spotted hyena optimizer [45]	Spotted hyena
2018	Lion pride optimization [46]	Lion pride behavior
2019	Seagull optimization algorithm [47]	Migration and attacking behaviors of a seagull
	Squirrel search algorithm [48]	foraging behavior of southern flying squirrels
	Pathfinder algorithm [49]	Bird flock
	Harris Hawks optimization [50]	Cooperative behavior and chasing style of Harris' hawks
	The Sailfish Optimizer [51]	Sailfish and Sardine swarm
	Sea Lion Optimization [52]	Sea lion

are seen engaging in their typical hunting activity, including a coordinated hunt, sharing of the catch, and a subsequent breakup into the smaller groups they first formed. Inter-pack hostility is uncommon because close-by packs frequently have relationships with one another[1].

Dholes hunt in groups, using coordinated tactics to surround their prey and gradually close in on them. They communicate and share information about the location of their prey with each other. Meanwhile, the prey employs strategies such as fleeing in the opposite direction, hiding, or seeking out the safest locations. The hunting behavior of dholes serves as the inspiration for an algorithm in which the predators are represented as a group of points in close proximity. During the hunting process, dholes give chase while the prey continuously seeks a better position. When the prey reaches an optimal position and cannot find a better one, the predator is considered to have won, indicating that the algorithm has achieved convergence.

Our suggested model assumes that the optimization domain is comparable to the plains and woods where dholes are looking for potential opportunities to hunt prey. Dholes do not know where to obtain food, which is similar to optimization challenges. The hunting grounds of the dholes might be harsh and foggy. As a result, it is challenging to approach and pursue the prey. Dholes need a specialized hunting method to increase their chances of success.

Dholes have adapted and developed an effective hunting mechanism, which we separate into two phases. Firstly, the pack hunting strategy is a local search mechanism, and the other is re-forming packs as a global search. Individuals in the super pack can share information about the domain during the global search phase (to exchange information about potential positions to hunt prey). On the other hand, in the local search, a pack depends on its search abilities. Therefore, our algorithm can calculate the different locations in the search domain. Besides, each individual and their neighbors observe the precision of the best possible position.

In contrast, we consider the location of prey to avoid being eaten as the optimal point of the set, where predators are difficult to reach. Prey are also sensitive when observing predators and always look for the best positions. As a result, the algorithm combines the optimal position of both the hunter and the prey. We also strategize to ensure the members in the pack keep their distance for the best chasing, avoiding falling early into one local minima location.

Those main inspirations are described and simulated as mathematical equations in the below sections.

B. INITIALIZATION

A population of N dholes are placed on the ground, randomly initialized as follows:

$$x_{i,j} = x_{i,j}^{min} + rand_{i,j} * (x_{i,j}^{max} - x_{i,j}^{min}) \qquad i = 1, \ldots N; j = 1, \ldots D \tag{1}$$

where $x_{i,j}$ is the random position vector of $i - th$ dhole with D dimensions, $x_{i,j}^{min}$, and $x_{i,j}^{max}$ denote the minimum and maximum values for the $j - th$ dimension of $i - th$ agent, and $rand$ is a uniform random value in the interval of [0, 1].

Let $f \in R^D$ be the criterion function of m variables according to the problem space dimensions. The x_{best} is the optimal solution (agent) if function $f(x_{best})$ value if global minimum or maximum. In DhoH, the position of prey is the optimal position the dhole tries to reach.

C. CLUSTERING-HUNTING BASED

This subsection describes the first step in the main DhoH algorithm. The algorithm is motivated by dholes' swarm hunting, ambushing tactics, and prey chasing. Moreover, they cannot all compete for each prey because of the large number of prey, so they will split into smaller groups to make hunting more efficient. To have a reasonable pack division, we base on the K-means algorithm to separate into many groups so that the members of each group tend to be close together. At the same time, the distance between the groups is as far as possible. Using K-means as an essential step in helping dhole have good distance separation. In other words, the algorithm makes it a point to maintain its distribution diversity. When the group members are close to each other, the probability of them falling into the same valley is higher, so the gradient approximation can be applied to compute a local minimum. In addition, it is possible to replace the K-means algorithm by selecting neighbors with the K-nearest neighbor selection algorithm. In our study, the K-means algorithm instead of KNN ensures uniform clustering in the optimal domain.

For the initial set with k random centroids $(c_1, c_2, \ldots, c_k)$, each observation should be assigned to the cluster with the mean closest to it using the least-squares error (Eq. 2). For each cluster allocated data, the update phase should recalculate the means centroids (Eq. 3).

$$C_i^{(t)} = \left\{ x_p : \left\| x_p - c_i^{(t)} \right\|_2 \leq \left\| x_p - c_j^{(t)} \right\|_2 \ \forall j, 1 \leq j \leq k \right\}, \tag{2}$$

$$c_i^{(t+1)} = \frac{1}{\left| C_i^{(t)} \right|} \sum_{x_j \in C_i^{(t)}} x_j \tag{3}$$

D. PACK HUNTING STRATEGY-MULTI-LOCAL SEARCH PHASE

This subsection describes the main algorithm, which is based on swarm interaction, acceleration, and capture strategies. Dholes move around their territory in search of potential prey as a herd of elk, caribou, and wild pigs. When prey is spotted, the pack starts to approach quietly from different directions as close as possible, trying to stay undetected. When they reach a good location, they run as fast as possible to attack by surprise. However, it will run in the opposite direction and try to find the locations where it feels safe that the dhole swarm can hardly find (local optimal point). Simultaneously, the pack members must surround the prey and keep a distance from other members.

For mathematical modeling, suppose the position of the prey is x_p, and the members of the dhole swarm are x_{d_i}, respectively ($2 \leq i \leq n_{members}$ where $n_{members}$ is the number of members in the dhole swarm). In the first step, the prey always has an advantage over the members of the dhole swarm.

The complete process of pack hunting is presented in Algorithm 1, where ∇v is seen as the gradient vector and $v_{support}$ helps guide the dholes to keep their distance from each other.

We can see that the graph $f(x)$ illustrated in Fig. 3 has more than one local optimal point. However, for close sets of locations, we would expect them to lie in the same valley. Besides, this algorithm can still work well even if several points are not in the same valley or are in the case of an imperfect valley. However, there is still one issue: how to group dholes into the same packs so that the corresponding dholes belong to the same valley. Since points in one group are usually close to each other, we expected that their location falls into a unique valley. In the chasing process, the dholes try to approach their prey while the prey runs away from all the dholes. The location of the dhole and prey are updated according to ∇v. However, the prey will be able to speed up and slow down depending on the terrain, so we use the kl_{target}

Algorithm 1 Pack Hunting Strategy

```
1:  Input: $x_d$ are the set of dholes' positions in the pack, and $x_p$ is the prey's position.
2:  Output: Find the best prey position $x_p^*$
3:  $kl_{target} \leftarrow 1.0$, $t \leftarrow 1$
4:  initialize $r_1$, $r_2$
5:  for $t \le n_{moves}$ do
6:      $i \leftarrow 1$
7:      for $i \le n_{members}$ do
8:          $\nabla v \leftarrow x_p - x_{d_i}$
9:          $x_p \leftarrow x_p + kl_{target} * \nabla v * \log|\nabla v|$
10:         $x_{d_i} \leftarrow x_{d_i} + random() * \nabla v$
11:         $j \leftarrow 1$
12:         for $j \le n_{members}$ do
13:             if $i \ne j$ then
14:                 $v_{support} \leftarrow x_{d_i} - x_{d_j}$
15:                 $x_{d_i} \leftarrow x_{d_i} - v_{support}$
16:         $i \leftarrow i + 1$
17:     update $kl_{target}$ using Eq. 4
18:     $t \leftarrow t + 1$
19: Change $kl_{mul}$ and $kl_{div}$ using $r_1$ and $r_2$
20: Return: $x_p$
```

coefficient to accelerate the convergence trend.

$$kl = \begin{cases} \dfrac{f(x_p^i) - f(x_p^{i-1})}{f(x_p^i) - \frac{(f(x_p^{i-2})+f(x_p^{i-3}))}{2}}, & \text{if } 2 * f(x_p^i) \\ & \ne (f(x_p^{i-2}) + f(x_p^{i-3})) \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

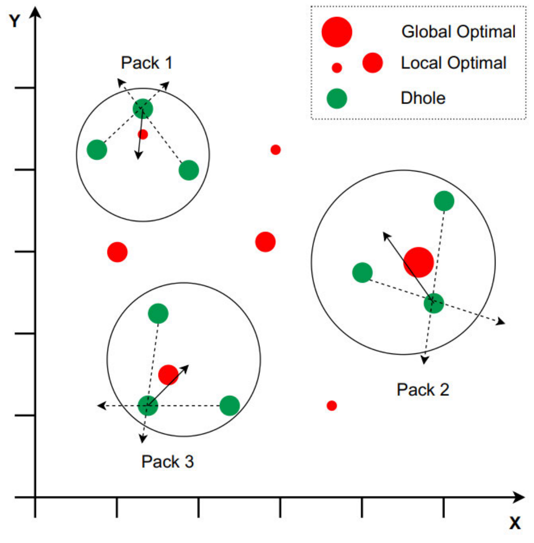
$$kl_{target} = \begin{cases} \dfrac{kl_{target}}{kl_{div}}, & \text{if } kl \le 1 \\ kl_{target} * kl_{mul}, & \text{otherwise} \end{cases} \tag{5}$$

$$kl_{mul} = kl_{mul} * r_1, \text{ if } kl \ge 1 \tag{6}$$

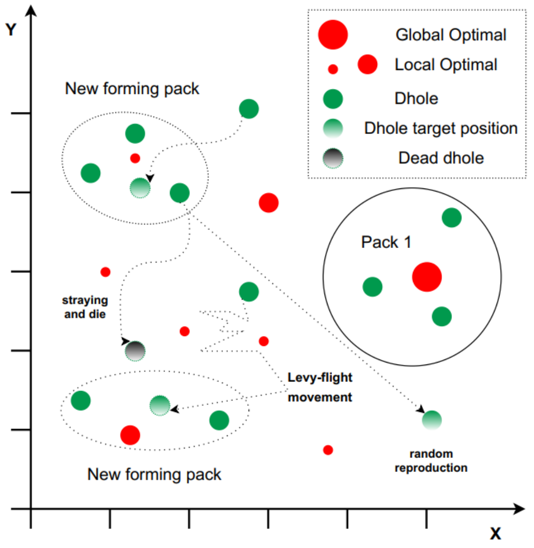
$$kl_{div} = kl_{div} * r_2, \text{ otherwise} \tag{7}$$

where $f(x_{p_i})$ is the fitness of dhole p in the i-th moves. kl_{div} is the coefficient decrease speed of divergence, and kl_{mul} is the coefficient increase speed of divergence. kl_{div} and kl_{mul} fluctuate from 1.0 to 2.0. n_{moves} is the number of moves in the pack-hunting strategy. It is different in the course of nature and thus different for the different optimization problems. After the packs catch their prey, they regroup into the super pack to share food and information about the local area.

In the hunting search phase, the algorithm adjusts the movement speed of dholes using the kl_{mul} and kl_{div} parameters, which can increase or decrease the speed appropriately to help dholes achieve local optimization quickly. Simultaneously, a probability parameter is determined to avoid getting stuck in small local areas. Moreover, two acceleration parameters (r_1 and r_2) are used to vary kl_{mul} and kl_{div}, respectively. The acceleration parameters (r_1 and r_2) help



(a) *Pack hunting strategy - local search phase*



(b) *Pack hunting strategy - global search phase*

FIGURE 2. Visualization of local and global search phase in DhoH.

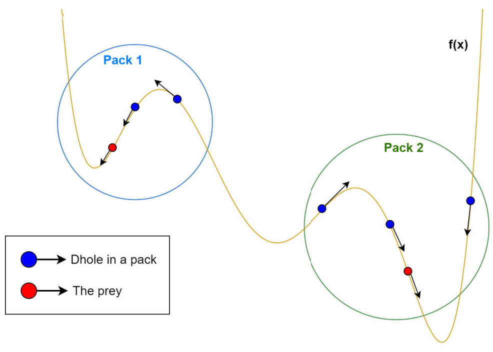


FIGURE 3. Illustration of movement of points in the same "valley" to reach the local minimum.

to adjust the velocity of the dhole to change more rapidly in areas of high variability, allowing for faster exploration of the search space. However, the sensitivity of these two parameters is quite large when changing the domain space, and there is not a single fixed value. Therefore, evaluating many different benchmarks makes it very difficult to select the fixed value of these two parameters. In this study, these two parameters (r_1 and r_2) are selected to be small and fixed to avoid too much deviation. The value of n_{moves} has a significant impact on the running time of the algorithm. However, n_{moves} does not necessarily need to be set to a very large value because the speed of the dhole can be increased exponentially by adjusting the numerical parameters r_1 and r_2. During testing, we use a fixed value for n_{moves} in all benchmark sets.

E. RE-FORMING PACKS-GLOBAL SEARCH PHASE

In forming packs and hunting together, there can be many conflicts among the dholes in the pack. Although they can stay close and communicate with each other, the food distribution issues make the habitat in the local area not as suitable as it was before. They may leave the pack and join another or re-form a new pack. In addition, dholes also face other dangers, such as being hunted by humans or other big predators like lions and leopards. Of course, if they cannot survive in the herd, they will have to find a way to reproduce (natural selection), adapt to the environment, or switch to another pack to live.

In DhoH, the weak dhole in the herd has to find a way to settle in another herd. If they successfully find a new pack, they can survive and join it to hunt and share food. If they can't find other flocks, they may become lost, have to hunt by themselves, or be attacked by other predators (or possibly die due to food shortage).

The successful re-forming pack process of dholes is described in Section II-E1. An unsuccessful re-forming pack process of dholes consists of two parts: the lost dholes have to go hunting alone, and the dholes starve or die by other predators described under Section II-E2. Fig. 2b) shows the visualization of the pack reforming process.

1) MOVING TO ANOTHER PACK

If a dhole successfully joins the new herd, it must adapt to the new environment and learn how to hunt prey from the new leader and other dholes in the new group. Therefore, our algorithm combines the features of three dholes, including a current dhole, a leader dhole, and a random strong dhole, to form the integration process.

$$x_d^{new} = x_d^{old} + \frac{\delta(x_{d_1}^{*} - x_d^{old})}{2} \times rand_1 + \frac{(1-\delta)x_{d_2}^{*}}{2} \times rand_2 \tag{8}$$

where x_d^{old} and x_d^{new} are the old and new positions of the chosen dhole, δ is a user-defined parameter, $x_{d_1}^{*}$ is the position of the leader dhole, and $x_{d_2}^{*}$ is the position of the strong dhole selected randomly from one of the K best dholes currently ($x_{d_1}^{*} \neq x_{d_2}^{*}$). $\mathbf{rand_1}$ and $\mathbf{rand_2}$ are random vectors with the same dimension as x, values in (0, 1).

2) STRAYING FROM THE PACK

The dhole encounters many obstacles when finding a new herd from its current herd. For example, bad weather or being chased by other predators causes them to stray from the pack. In addition, dholes can also die from lack of food on the way, especially old dholes that do not have enough strength to move.

- For those dholes killed by the other predator or due to natural selection in searching for a new herd, a random dhole in the environment will reproduce and replace the dead one using Eq. 1.
- Meanwhile, dholes that survive being hunted by other predators or natural selection must possess unique characteristics and skills to adapt to new environments and the process of hunting alone. To simulate the foraging process of these dholes, we use the Levy-flight technique.

In the ***Levy-flight trajectory***, the step length follows the heavy-tailed Levy distribution to model the foraging trajectory of dholes [58]. Numerous researchers have demonstrated that many creatures, including birds, insects, and marine animals, follow the mathematical Levy distribution when feeding. These studies formed the Levy flight foraging hypothesis: Levy-flight can improve the efficiency and accuracy of natural foraging and is more naturally adaptable. As a global searching operator, the Levy-flight trajectory searches for space using short-distance walking and long-distance jumping routes. Those two abilities help improve the population's diversity and local exploitation ability, especially with the approximate formula proposed by Mantegna [59], which generates random numbers obeying the Levy distribution. In general, Levy step size can be expressed as:

$$Levy(s) \sim |s|^{-1-\beta} \; with \; 0 < \beta \leq 2 \tag{9}$$

$$s = \frac{\mu}{|v|^{1/\beta}}, \; \mu \sim N(0, \sigma_\mu^2), \; v \sim N(0, \sigma_v^2) \tag{10}$$

$$\sigma_\mu = \left[\frac{\Gamma(1+\beta) \times \sin(\pi.\beta/2)}{\Gamma((1+\beta)/2) \times \beta \times 2^{(\beta-1)/2}}\right]^{1/\beta}, \; \sigma_v = 1 \tag{11}$$

where s is the step length of the Levy flight calculated by Mantegna's algorithm, μ and v are chosen from the normal distribution, β in (0, 2], and Γ is a gamma function.

$$x_d^{new} = Levy(x_d^{old}) \tag{12}$$

where x_d^{old} and x_d^{new} are the previous position and the current generated position of the current dhole.

3) EXPLORATION AND EXPLOITATION RATE

In the DhoH algorithm, four coefficients help the exploration and exploitation process and balance those two processes. α is calculated by Eq. 13 and is used to calculate δ (Eq. 14) and θ (Eq. 16). δ is a parameter mentioned in Eq. 8. Meanwhile,

θ and γ (Eq. 15) are used to discount the convergence speed.

$$\alpha = arctanh(-(\frac{g+1}{g_{max}} + 1)) \tag{13}$$

$$\delta = \frac{exp(\alpha)}{exp(arctanh(\frac{g_{max}-1}{t_{max}})}) \tag{14}$$

$$\gamma = (1 - \frac{g+1}{g_{max}}) \times cos(\frac{\pi}{3} \times \frac{g+1}{g_{max}}) \tag{15}$$

$$\theta = \frac{(1 - \frac{g+1}{g_{max}})}{2} \times exp(\frac{\alpha}{2}) \tag{16}$$

where g is the current generation (iteration/epoch), and g_{max} is the maximum number of generations. For termination with the maximum number of function evaluations, g and g_{max} are replaced by current evaluation count fes and maximum evaluation bound fes_{max}.

Algorithm 2 Dholes Hunting-Based Optimization (DhoH)

```
1:  Input: $g_{max}$ is the maximum number of epochs, $t_{max}$ is the time-bound, population $P$ with $N$ dholes randomly,
2:  Output: The global best solution $X_{best}$
3:  while not termination do
4:      Shuffle the population
5:      Split the population into the $G$ (set of the packs) using the K-means algorithm
6:      $n \leftarrow$ G size
7:      $P' \leftarrow \emptyset$
8:      i ← 1
9:      for $i \leq n$ do
10:         $x_{new} \leftarrow$ hunting strategy (Alg 1)
11:         Add $x_{new}$ into $P'$
12:         i ← i + 1
13:     Update $\gamma, \theta, \delta$
14:     while $|P'| < N$ do
15:         if rand() $< \gamma$ then
16:             if rand() $< \theta$ then
17:                 $x_{new} \leftarrow$ generate a dhole and add into $P'$ using Eq. 1
18:             else
19:                 $x_{old} \leftarrow$ random dhole from $P$
20:                 $x_{new} \leftarrow$ find a new dhole using Levy-flight technique (Eq. 12)
21:         else
22:             $x_{old} \leftarrow$ randomly select one of the best K dholes in the population $P'$
23:             $x_{new} \leftarrow$ find a new dhole using $x_{old}$ and global best from Eq. 8
24:     Update the global best solution $X_{best}$
25:     $P \leftarrow P'$
26: Return: $X_{best}$
```

4) ALGORITHM COMPLEXITY

With the assumption that we have a population with N dholes, D is the number of dimensions of the problem, the maximum number of generations is g_{max}, and the cost for the fitness function is C. The proposed algorithm uses n_{moves} moving steps to find a local best solution in the pack hunting strategy. The number of function evaluations in this step is $\frac{N*n_{move}}{G_{size}}$. In the re-forming pack phase, the DhoH algorithm only uses common transforming operators. The number of evaluations in this step is $2N/G_{size}$. Consequently, the average complexity of the proposed algorithm is estimated as $O(g_{max} * N * n_{move} * C)$.

III. EXPERIMENTAL STUDY

A. PARAMETER TUNING AND THEORETICAL EXPERIMENT

This part comprises both the first experiments: parameter tuning to decide the best setup for DhoH and prove its effectiveness in comparison with state-of-the-art methods. The experiments use benchmark functions from the CEC2017 test suite [60] with 10 dimensions to test the proposed algorithm. DhoH's performance is evaluated with different values for its three key parameters: the number of local searches n_{moves} and the two acceleration parameters r_1 and r_2. The results of DhoH with the best parameter set are also compared with all seven algorithms mentioned above.

1) BENCHMARK FUNCTIONS

The performance of DhoH has been experimented on 29 benchmark functions (except f_2, which has been deprecated from the official suite). They are divided into four groups of functions:

- 2 unimodal functions (f_1 and f_3): they have only one global optimal point in the search space.
- 7 multimodal functions ($f_4 - f_{10}$): they have one global optimal point going along with several local minimum points.
- 10 hybrid functions ($f_{11} - f_{20}$): the variables are randomly divided into sub-components, and then different basic unimodal and multimodal functions are used for different sub-components.
- 10 composition functions ($f_{21} - f_{30}$): they merge the properties of the sub-functions better and maintain continuity around the global/local optima.

The optimal value of each function f_i is $f_i^* = 100 \times f_i$. For The details of functions, characteristics can be found in [60]. The 3D plots of several benchmark functions are presented in Fig. 4.

2) PERFORMANCE METRICS AND TEST CASES

The experimental results of each model are produced by calculating the mean (Eq. 17) and standard deviation *std* (Eq. 18) of 10 running times.

$$mean = \frac{1}{N}\sum_{j=1}^{N} f(x^j) \tag{17}$$

$$std = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(f(x^j) - \mu)^2} \tag{18}$$

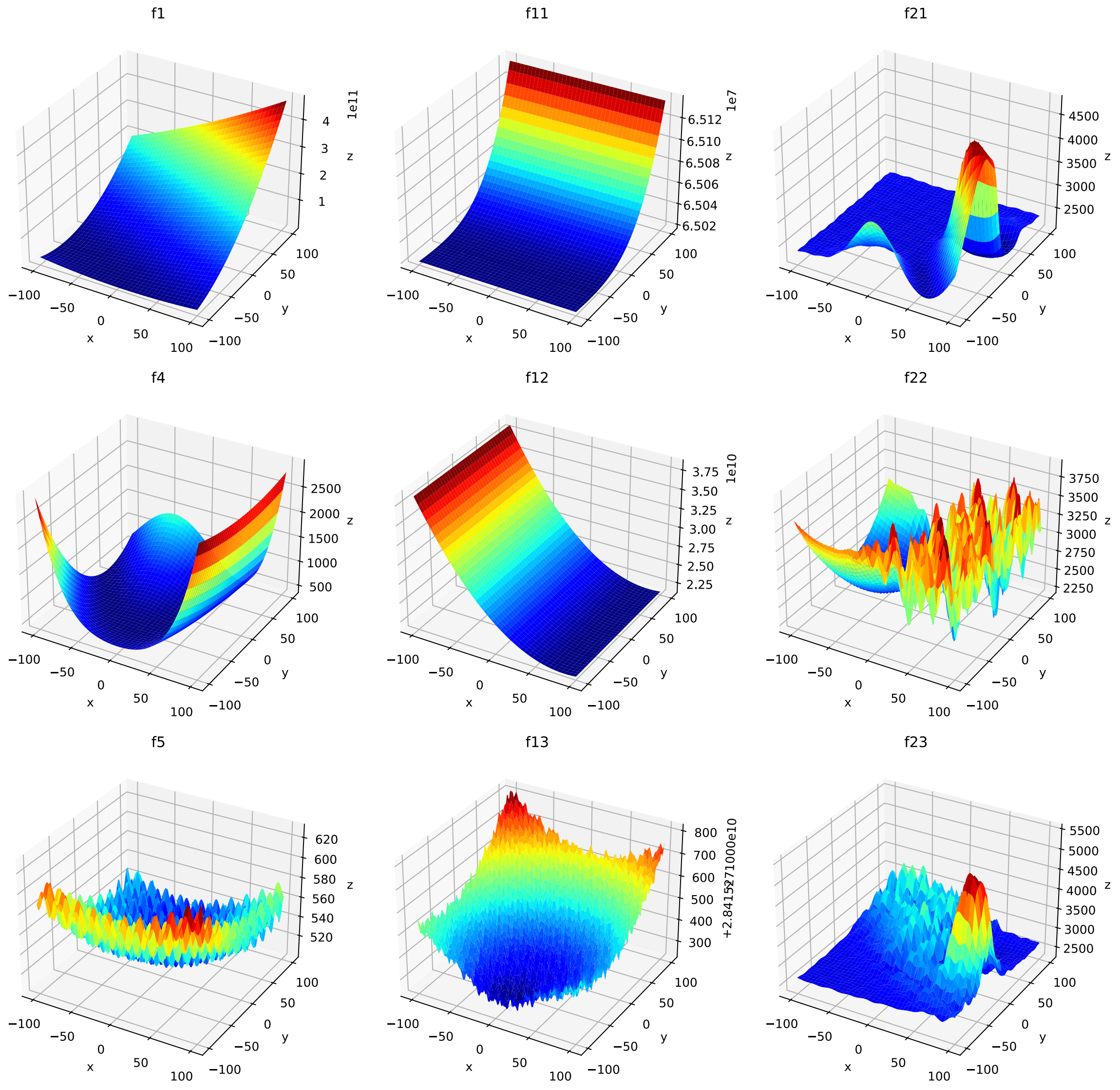


FIGURE 4. 3D illustrations of some benchmark functions.

where $N = 19$ is the size of the observed population, $f(x^j)$ $(j = 1, 2, \ldots, N)$ are observations, and μ is the population mean. After calculating the *mean* and *std* values for each function and each algorithm, the best results will be highlighted in both. The following rules determine the best results:

- *Mean* values are considered first. If, in a case, an algorithm owns the best *mean* value, it will be ranked as the best optimizer.
- In the case where two or more algorithms have the same *mean* value, the one that has the most stable *std* value will be chosen as the best one.

3) PARAMETER SETTINGS

In this experiment, the focus is on tuning three important parameters, n_{moves}, r_1, and r_2, to improve the performance of the simulated annealing algorithm. There are two sub-experiments.

The first sub-experiment aims to estimate the optimal value of n_{moves}, which determines the number of iterations or moves that the algorithm makes at each temperature level. The other two parameters, r_1 and r_2, are fixed to 1.25 and 1.05, respectively. Three different values of n_{moves}, namely 3, 5, and 7, are tested to determine which value produces the best results.

In the second sub-experiment, the goal is to tune the values of r_1 and r_2 for a fixed value of n_{moves}. In this case, n_{moves} is set to 5, and a parameter grid of 16 combinations of r_1 and r_2, with values of 1.05, 1.15, 1.25, and 1.5, is evaluated to find the best combination that maximizes the algorithm's performance.

4) ALGORITHMS FOR EXPERIMENTATION

After selecting the optimal settings for DHoH, it was compared with ten recent metaheuristic algorithms listed below. Note that some of these algorithms were implemented using the MEALPY Python library [61].

- A more modern variation of the well-known whale optimization technique is the hybrid improved whale optimization algorithm (HI-WOA [62]). In order to increase variety and convergence speed, a novel feedback system with a nonlinear convergence factor and inertia weight coefficient is introduced.
- The Artificial Bee Colony (ABC [63]) is an algorithm proposed by Karaboga and Akay The algorithm describes the behavior of bees searching for honey with an effective strategy to optimize a large set of numerical test functions.
- Biogeography-Based Optimization (BBO [64]) involves examining how biological organisms are distributed geographically. The algorithm is inspired by the idea that we can gain insights from nature, leading to the use of biogeography in solving optimization problems.
- A new version of the genetic algorithm known as the enhanced real-coded genetic algorithm (RCGA-rdn [65]) aims to increase searchability by incorporating three newly created operators: ranking group selection, direction-based crossover, and normal mutation. Both challenging numerical puzzles and real-world engineering issues have demonstrated the viability of this technique.
- The gaining-sharing knowledge-based algorithm (GSKA [66]) is based on the idea that information is acquired and shared during human life, with two key stages: junior and senior. Its solutions in continuous optimization problems are competitive.
- The adaptive equilibrium optimizer (A-EO [67]), which implements adaptive decision-making of dispersal for search agents who do not perform well, enhances the standard equilibrium optimizer. It has been said that it outperforms many cutting-edge algorithms.
- Successful history-based adaptive differential evolution with linear population size reduction (L-SHADE [68]) is a variant of DE [69], renowned for its exceptional aptitude for numerical function optimization. As implied by its name, L-SHADE employs two noteworthy mechanisms: productive history-based adaptation and linear population size reduction. L-SHADE was the winner of CEC2014.
- jSO [70] is an improved variant of L-SHADE with several parameter control tweaks and a modified mutation operator. jSO was also the runner-up of CEC2017.
- An optimization technique called the slime mould algorithm (SMA [71]) is based on the slime mold's foraging and diffusion behavior. It provides a special model that simulates feedback processes using adaptive weights and is helpful for both exploration and exploitation.
- Improved Grey Wolf Optimizer (I-GWO [72]) is an enhanced version of Grey Wolf Optimizer (GWO [34]), a nature-inspired optimization algorithm that mimics the hunting behavior of grey wolves to solve optimization problems. It introduces modifications to the original algorithm, such as a new position update equation and a modified search mechanism, to improve its efficiency and performance in finding the global optimum solution.

These algorithms, along with DHoH, were evaluated in two additional experiments. The first experiment focused on theoretical evaluations, utilizing 29 benchmark functions from the CEC2017 competition. The second experiment concentrated on real-world performance evaluations, aiming to optimize the decentralized level of a modern blockchain protocol.

To ensure fairness in the running time of the different algorithms, we set the maximum number of function evaluations at $nfes_{max} = 1000000$. For all algorithms, a population size $p_s = 100$ was used for each function within a 10-dimensional search space. The parameter choices were based on the setups described in the original papers of each algorithm. For our proposed DHoH algorithm, the parameter values that yielded the best results were selected after an initial parameter tuning experiment.

The code of this study is available at[2] [73].

B. PRACTICAL EXPERIMENT

In this experiment, we apply the proposed DhoH algorithm to solve a challenging optimization problem in blockchain: decentralized level optimization for the recent consensus protocol named Meta-heuristic Proof of Criteria (MPoC) proposed in [56]. First, we provide insights into consensus protocols, especially MPoC, and the importance of the decentralized level within this context.

Blockchain, introduced by Satoshi Nakamoto in 2008, is a decentralized, tamper-resistant ledger recording transactions across a distributed network of computers, ensuring transparency, security, and immutability [74]. Consensus protocols, crucial in blockchain ecosystems, enable agreement among network participants on transaction validity and ledger state. From Proof of Work (PoW [74]) to Proof of Stake (PoS [75]), these protocols establish trust in a trustless environment. PoW relies on miners solving cryptographic puzzles, consuming computational power, while PoS uses validators staking cryptocurrency holdings as collateral. Recently, many blockchain platforms have adopted Delegated Proof-of-Stake (DPoS DPoS [76]). DPoS builds on PoS by allowing token holders to vote for a limited number of delegates to validate transactions and produce blocks, enhancing scalability and efficiency. However, DPoS tends to centralize block production to a few outstanding nodes, as network participants often vote for nodes likely to become block producers.

The recent consensus protocol MPoC [56] has effectively addressed centralization issues and overcome the disadvantages of DPoS. The authors introduced a novel "decentralized level" metric that enables multiple operational criteria to

[2]https://github.com/NeiH4207/DhoH

evaluate all blockchain nodes during the block producer selection. MPoC's scheme is as follows: criteria are assigned unique weight factors and aggregated to formulate a final objective value to evaluate blockchain nodes and choose the best block producer for each transaction. For each set of weight factors, the decentralized level measures the democracy achieved during the selection process of block producers. This metric is assessed based on two key factors: the number of nodes becoming block producers and the variance in the occurrence of all nodes becoming block producers in specific rounds. A higher number of nodes assuming block producer roles signifies greater network democracy.

Consequently, to achieve optimal democracy in the blockchain network, MPoC addresses an optimization problem aimed at maximizing the decentralized level:

$$W^* = \arg\max_{W}(decentralized\_level) \tag{19}$$

$$decentralized\_level = 1 - \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(\tilde{N}_i - \overline{N})^2} \tag{20}$$

$$\tilde{N}_i = \frac{N_i}{\max_{[N_1, N_2, \ldots, N_m]}} \tag{21}$$

$$criteria\_total\_value = W * criteria = \sum_{i=1}^{h} w_i * criteria_i \tag{22}$$

where $\overline{N}$ is the mean value of $[\tilde{N_1}, \tilde{N_2}, \ldots, \tilde{N_m}]$. m is the number of blockchain nodes, and N_i is the number of times that node i became block producers. The vector $[N_1, N_2, \ldots, N_m]$ is calculated by the blockchain simulator. W is a vector of h-dimensions $[w_1, w_2, \ldots, w_h]$, where h is the number of criteria, and w_i is the weights of i^{th} criterion. Note that a node with a higher *criteria_total_value* than others indicates that it is more worthy of becoming a block producer than others. Subsequently, an optimization algorithm is applied to find the optimal set of criteria weights that maximize the decentralized level of the node selection process, thereby enhancing the democracy of the blockchain network.

The block producer selection process of MPoC is based on various operational factors to evaluate all blockchain nodes, rather than relying on a single staked value as in DPoS. This allows blockchain networks using MPoC to design their block producer selection strategies without compromising the network's security. However, this introduces a new challenge: balancing criteria quality and optimization complexity. While more diverse and reliable criteria improve the quality of the block producer selection process, they also increase the scale of the optimization problem. Selecting an effective optimization algorithm that can handle a significantly increased number of factors is a top priority.

To address this, the practical experiment was conducted to evaluate the performance of DHoH and other methods (HI-WOA, GSKA, L-SHADE, SMA and A-EO) on this specific problem, demonstrating DHoH's capability in real-world applications.

TABLE 2. Noteworthy performance results of DhoH with different parameter settings on CEC2017 benchmark functions. ***The bold values indicate outstanding mean values for each function.***

Parameters			Functions		
n_{moves}	r_1	r_2	f_1	f_{12}	f_{30}
1			2.38e+3	1.07e+4	**4.22e+3**
3			7.71e+2	8.19e+3	5.37e+3
5	1.25	1.05	**3.68e+2**	**5.70e+3**	5.12e+3
7			**3.86e+2**	**5.30e+3**	4.98e+3
9			5.27e+2	8.67e+3	6.82e+3
	1.05	1.05	366.5716	6882.127	4985.569
	1.05	1.15	333.8631	7451.478	4678.067
	1.05	1.25	551.5745	5782.427	4481.666
	1.05	1.5	300.4071	5460.119	4652.441
	1.15	1.05	**280.834**	**5385.563**	5343.708
	1.15	1.15	857.4707	6576.333	4525.55
	1.15	1.25	599.68	7653.56	4954.907
	1.15	1.5	702.2136	9415.855	5231.085
5	1.25	1.05	1141.585	8571.877	4590.982
	1.25	1.15	437.5991	9029.286	5799.029
	1.25	1.25	815.1149	7869.986	4692.097
	1.25	1.5	571.0378	6321.427	4749.387
	1.5	1.05	2199.066	7677.58	**4285.582**
	1.5	1.15	753.5948	7295.486	6640.046
	1.5	1.25	2561.902	6303.952	6061.948
	1.5	1.5	437.483	7945.876	5856.669

IV. RESULTS AND DISCUSSION

In this section, the impact of some parameters on DhoH's effectiveness is discussed. After that, we show the results of the DhoH algorithm against various well-known algorithms and discuss some critical points among the compared algorithm's performance on both the benchmark function tests and practical tests.

The algorithm model utilizes strategies for discovering, exploiting, and maintaining population diversity to optimize its performance. Analyzing the effect of each component through experimental results is complicated as all strategies are combined with each other. However, for the discovery strategy, the parameter N (number of dholes in the population) can bring different effects depending on the complexity and size of the spatial domain. The larger and more complex the domain, the more dholes will be needed to save the model's discovery time in the early stages. However, the number of dholes does not affect the convergence rate of the algorithm on each epoch after the model has found good local optimal points. Additionally, the discovery process can be observed during hunting when dhole groups in different regions interact randomly with each other.

For the discovery process, the model uses a gradient approximation optimization approach based on information about groups of dholes that are close to each other. Local optimal points are quickly found as soon as dholes and prey fall into these areas. The speed of this search process is influenced by two parameters, klmul and kldiv, which help to effectively combine mining discovery and increase or decrease the moving speed of the dhole when the slope of

TABLE 3. Mean results for DhoH and other algorithms on CEC2017 benchmark functions. ***The bold values indicate the best mean values for each function.***

Function		HI-WOA	RCGA-rdn	GSKA	L-SHADE	SMA	A-EO	I-GWO	jSO	ABC	BBO	DhoH
Unimodal	f_1	7.56e+10	1.17e+9	8.83e+2	**1.00e+2**	1.01e+4	4.61e+3	6.30e+10	1.52e+2	5.08e+8	7.23e+8	3.42e+2
	f_3	1.14e+4	1.13e+3	**3.00e+2**	**3.00e+2**	**3.00e+2**	**3.00e+2**	1.39e+4	**3.00e+2**	1.80e+4	1.44e+4	**3.00e+2**
Multimodal	f_4	8.97e+2	4.13e+2	**4.00e+2**	**4.00e+2**	4.01e+2	4.02e+2	9.11e+2	4.05e+2	4.08e+2	4.09e+2	**4.00e+2**
	f_5	5.92e+2	5.32e+2	5.18e+2	**5.02e+2**	5.14e+2	5.28e+2	5.74e+2	5.03e+2	5.50e+2	5.22e+2	5.07e+2
	f_6	6.57e+2	6.09e+2	**6.00e+2**	**6.00e+2**	**6.00e+2**	6.28e+2	6.51e+2	**6.00e+2**	6.14e+2	6.11e+2	6.01e+2
	f_7	8.13e+2	7.66e+2	7.27e+2	**7.12e+2**	7.23e+2	7.53e+2	8.09e+2	7.13e+2	7.71e+2	7.53e+2	7.21e+2
	f_8	8.52e+2	8.31e+2	8.17e+2	**8.02e+2**	8.09e+2	8.18e+2	8.51e+2	8.04e+2	8.60e+2	8.27e+2	8.13e+2
	f_9	1.65e+3	9.48e+2	**9.00e+2**	**9.00e+2**	**9.00e+2**	1.13e+3	1.43e+3	**9.00e+2**	1.01e+3	9.73e+2	**9.00e+2**
	f_{10}	2.66e+3	1.86e+3	1.71e+3	**1.02e+3**	1.52e+3	1.89e+3	2.77e+3	1.22e+3	2.70e+3	1.76e+3	1.33e+3
Hybrid	f_{11}	2.61e+3	1.14e+3	**1.10e+3**	**1.10e+3**	1.11e+3	1.14e+3	3.55e+3	**1.10e+3**	1.17e+3	1.32e+3	1.11e+3
	f_{12}	4.23e+8	1.06e+7	3.23e+6	**1.20e+3**	3.58e+4	1.99e+4	1.07e+9	1.34e+3	2.43e+7	4.47e+7	9.39e+3
	f_{13}	7.87e+5	2.09e+4	4.29e+3	**1.30e+3**	1.91e+4	9.20e+3	7.55e+6	**1.30e+3**	5.52e+4	1.02e+7	1.62e+3
	f_{14}	4.04e+3	1.52e+3	1.42e+3	**1.40e+3**	1.44e+3	1.53e+3	5.69e+3	**1.40e+3**	2.62e+3	6.06e+4	1.44e+3
	f_{15}	1.55e+4	1.74e+3	1.59e+3	**1.50e+3**	1.52e+3	2.23e+3	2.23e+4	**1.50e+3**	9.86e+3	1.77e+5	1.53e+3
	f_{16}	2.14e+3	1.63e+3	1.61e+3	**1.60e+3**	1.64e+3	1.83e+3	2.00e+3	**1.60e+3**	1.86e+3	1.83e+3	**1.60e+3**
	f_{17}	1.83e+3	1.75e+3	1.73e+3	**1.70e+3**	1.74e+3	1.75e+3	1.83e+3	1.71e+3	1.82e+3	1.77e+3	1.73e+3
	f_{18}	1.25e+5	1.69e+4	1.82e+3	**1.81e+3**	1.51e+4	4.94e+3	2.34e+6	**1.81e+3**	2.10e+5	4.64e+5	1.90e+3
	f_{19}	4.54e+6	2.15e+3	**1.90e+3**	**1.90e+3**	1.91e+3	6.39e+3	2.07e+5	**1.90e+3**	1.31e+4	5.02e+5	1.91e+3
	f_{20}	2.18e+3	2.06e+3	**2.00e+3**	2.01e+3	2.03e+3	2.08e+3	2.24e+3	2.01e+3	2.16e+3	2.04e+3	2.03e+3
Composition	f_{21}	2.37e+3	2.21e+3	2.28e+3	2.23e+3	2.20e+3	2.29e+3	2.35e+3	2.23e+3	2.32e+3	2.27e+3	**2.19e+3**
	f_{22}	2.70e+3	2.32e+3	2.30e+3	2.30e+3	2.29e+3	2.34e+3	2.83e+3	2.30e+3	2.32e+3	2.30e+3	**2.20e+3**
	f_{23}	2.72e+3	2.60e+3	2.62e+3	2.61e+3	2.62e+3	2.65e+3	2.71e+3	2.60e+3	2.65e+3	2.63e+3	**2.53e+3**
	f_{24}	2.85e+3	2.53e+3	2.74e+3	2.71e+3	2.52e+3	2.73e+3	2.85e+3	2.69e+3	2.76e+3	2.77e+3	**2.47e+3**
	f_{25}	3.25e+3	2.95e+3	2.93e+3	2.92e+3	2.92e+3	2.92e+3	3.17e+3	2.91e+3	2.95e+3	2.94e+3	**2.90e+3**
	f_{26}	3.77e+3	2.99e+3	2.90e+3	2.90e+3	2.94e+3	3.21e+3	3.92e+3	2.90e+3	3.18e+3	2.98e+3	**2.68e+3**
	f_{27}	3.19e+3	3.09e+3	3.09e+3	3.07e+3	3.09e+3	3.08e+3	3.20e+3	3.09e+3	3.10e+3	3.10e+3	**3.06e+3**
	f_{28}	3.30e+3	3.19e+3	3.15e+3	3.27e+3	3.10e+3	3.21e+3	3.48e+3	3.20e+3	3.35e+3	3.26e+3	**3.04e+3**
	f_{29}	3.46e+3	3.17e+3	3.16e+3	**3.13e+3**	3.15e+3	3.29e+3	3.40e+3	3.14e+3	3.29e+3	3.23e+3	3.14e+3
	f_{30}	1.01e+7	1.13e+5	6.44e+4	**3.20e+3**	5.68e+3	4.01e+3	2.65e+7	3.85e+3	6.70e+6	2.65e+6	4.54e+3

TABLE 4. Statistical results of Wilcoxon signed rank test for DhoH versus other algorithms.

Result	HI-WOA vs DhoH	GSKA vs DhoH	RCGA-rdn vs DhoH	jSO vs DhoH	ABC vs DhoH
Ranks (-/+/=)	0/29/0	0/29/0	6/19/4	16/9/4	0/29/0
Z	-4.703	-4.706	-3.176	-0.902	-4.703
(based on)	(negative ranks)	(negative ranks)	(negative ranks)	(positive ranks)	(negative ranks)
p (2-tailed)	0.000	0.000	0.001	0.367	0.000
Result	L-SHADE vs DhoH	SMA vs DhoH	A-EO vs DhoH	I-GWO vs DhoH	BBO vs DhoH
Ranks (-/+/=)	17/8/4	3/20/6	1/27/1	0/29/0	0/29/0
Z	-1.212	-3.805	-4.157	-4.703	-4.703
(based on)	(positive ranks)	(negative ranks)	(negative ranks)	(negative ranks)	(negative ranks)
p (2-tailed)	0.226	0.000	0.000	0.000	0.000

the search area changes. Moreover, a probability parameter is used to enable dhole to accept changes in position and cross saddle points easily.

A. PARAMETER TUNING

The results of DhoH with different settings for three parameters n_{moves}, r_1, and r_2 are summarized in Table 2. We have reported results for only three specific functions, namely f_1, f_{12}, and f_{30}, as these are the only functions that exhibit significant differences in results across DhoH's settings. In contrast, the performance of the other functions remains largely unchanged across all settings.

In the first part of the table, with fixed parameters $r_1 = 1.25$ and $r_2 = 1.05$, the optimal values for n_{moves} are either 5 or 7. An interesting observation is that these points form an optimal range [5, 7] within the tested range of n_{moves}. Therefore, we choose $n_{moves} = 5$ to represent this range in subsequent experiments.

However, in the second part, with $n_{moves} = 5$, there is no significant difference among the results achieved by the 16 distinct value pairs of (r_1, r_2). The pair $(r_1, r_2) = (1.15, 1.05)$, which yields the most positive results, is selected. As a consequence, the chosen setting for DhoH in later experiments is $n_{moves} = 5$, $r_1 = 1.15$, and $r_2 = 1.05$.

B. THEORETICAL EXPERIMENT

As described in section III-A, we compare the results of DhoH with seven existing methods to prove the proposed algorithm's effectiveness. The average results on the benchmark set are observed and verified with statistical tests. The convergence trend and stability of the mentioned methods are also analyzed.

1) GENERAL PERFORMANCE

Table 3 summarizes the mean values obtained from 10 consecutive runs of each considered algorithm on each benchmark function. In the Unimodal set, many algorithms

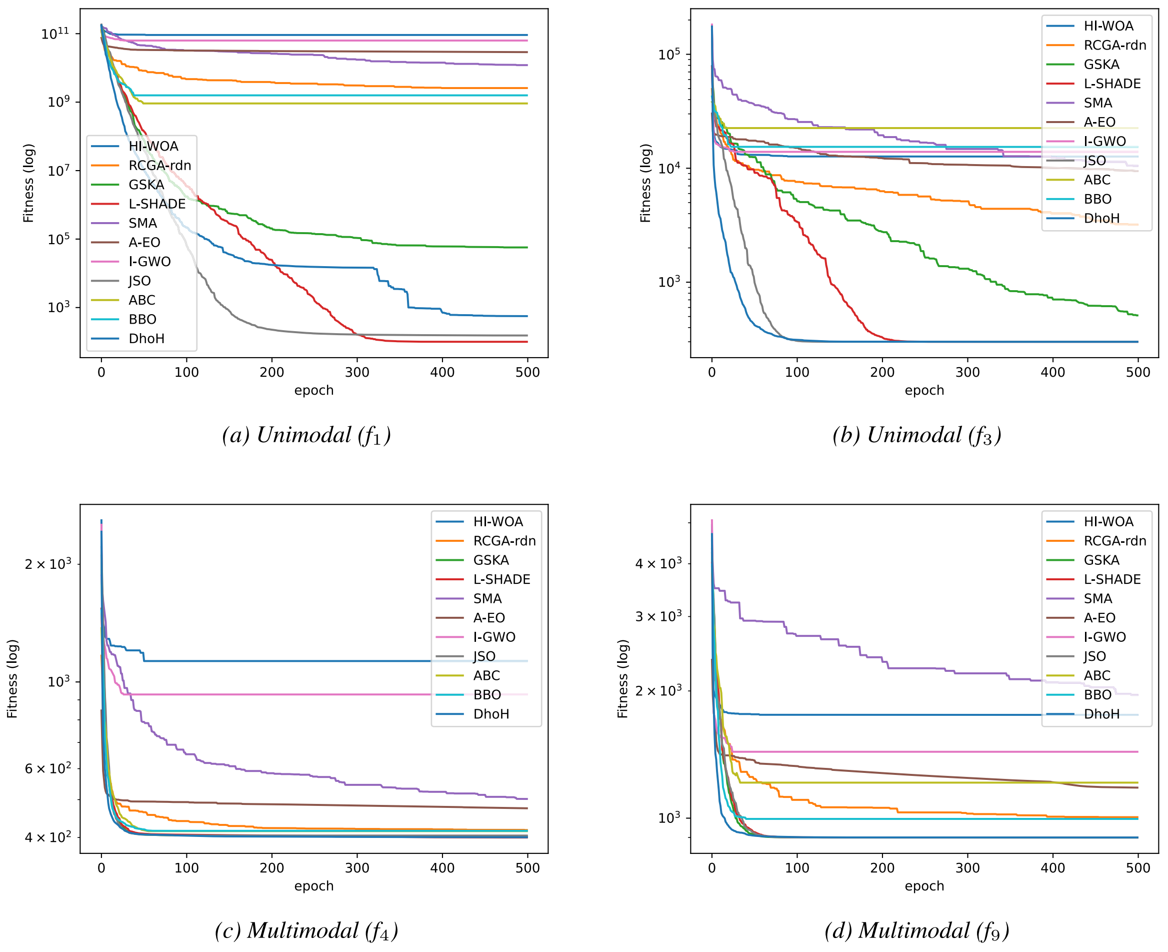
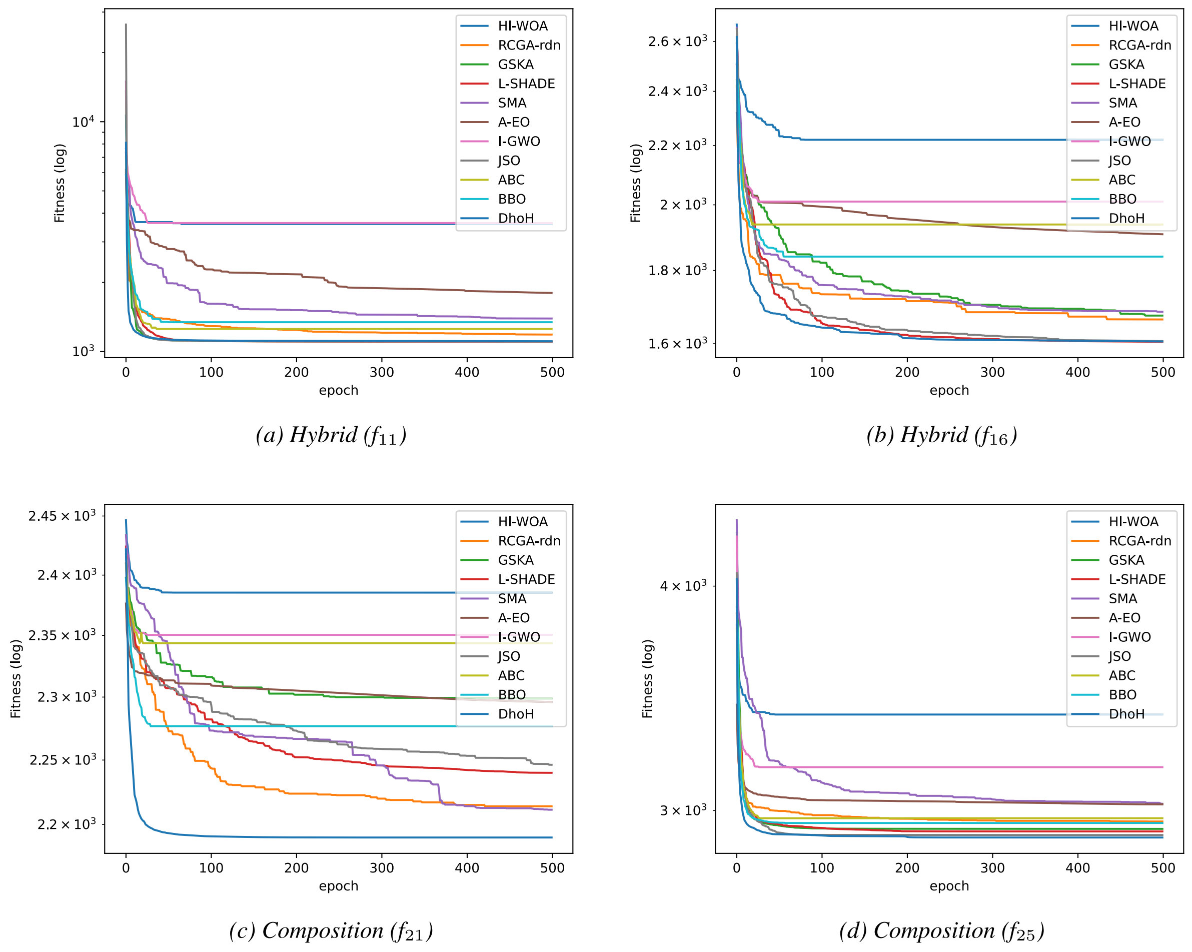


FIGURE 5. **Convergence trend of DhoH and other algorithms on unimodal and multimodal functions.**

can reach the optimal point, especially HI-WOA, which reaches this point in a short time. However, the proposed algorithm can not accomplish this as quickly, partly because the running time is not fast enough, but the main reason is that the algorithm prioritizes discovery rather than diving into a single local optimal point. On other complex functions, most of the methods fail to reach good results. In particular, HI-WOA and A-EO do not show good performance enough.

The DhoH algorithm is designed to find the optimal solution in the most complex space domain. The algorithm always maintains the diversity of the population by keeping the maximum distance between clusters in K-means and generating new random Dhole individuals in unexplored areas. In addition, the algorithm converges very quickly at local optimal points and can overcome bad local optimal points due to the probability parameter for selecting movements. The experimental results show that the algorithm achieved the best results for benchmark sets with high dimensions and complex domains (5/5 composition sets). In detail, L-SHADE, jSO, and the proposed DhoH are the most remarkable in most complex cases. L-SHADE, jSO, and DhoH have the best results on 10/30, 21/30, and 12/30 functions, respectively. Since composition is the most complicated type of benchmark, these results show that the proposed method is suitable for highly complex functions. It is worth noting that almost all other algorithms are stuck at local optima, while the DhoH algorithm can find more optimal points, even the most optimal point with high stability across 10 different runs.

We also conducted a Wilcoxon signed-rank test on the results provided in Table 3 to verify the performance of DhoH. With a significance level of $\alpha = 0.05$, this statistical hypothesis test can check whether the proposed DhoH is better than other methods. Since we perform seven comparison tests, Bonferroni Correction should be applied. Thus, the new significance level is $0.05/10 = 0.005$. Statistical results are collected in Table 4. This table shows that DhoH outperforms eight methods: HI-WOA, GSKA, RCGA-rdn, SMA, A-EO, I-GWO, ABC, and BBO. Their corresponding p-values are smaller than 0.006, and the comparison ranks favor DhoH. On the contrary, the p-values for the pairs DhoH - L-SHADE and DhoH - jSO are bigger ($0.226 > 0.006$ and $0.367 > 0.006$). This means there is no significant difference between the results of DhoH, jSO,



(a) Hybrid (f_{11}) *(b) Hybrid (f_{16})*

(c) Composition (f_{21}) *(d) Composition (f_{25})*

FIGURE 6. Convergence trend of DhoH and other algorithms on hybrid and composition functions.

and L-SHADE. Since the proposed algorithm outperforms multiple existing methods and almost equals L-SHADE and jSO, we can conclude that DhoH works effectively in optimizing benchmark functions.

2) CONVERGENCE TREND

In this subsection, we visualize the convergence process of each algorithm in 500 iterations. Fig. 5 and Fig. 6 visualize the convergence trend during one run to optimize some benchmark functions. We notice that the convergence trend of algorithms has different characteristics depending on the benchmark type. Since unimodal functions such as f_3 have smooth and simple shapes, all methods tend to converge to optimal solutions early. Surprisingly, f_1 proves to be a difficult challenge as only L-SHADE comes close to the optimal value. The same can be said about hybrid ones since a major part of their components is unimodal functions. In contrast, most methods have trouble with multimodal functions (which have a vast number of local optima). As shown in f_4 and f_9, L-SHADE and the proposed DhoH have the best result, which means it is better at escaping local optima. The same challenge can be in hybrid functions, which combine the traits of different sub-functions. Thus, only L-SHADE and DhoH can give decent results for f_{11} and f_{16}. Similarly, the asymmetrical and complex form of composition functions f_{21} and f_{25} make them difficult to optimize. The two last sub-figures give the clearest view of the distinctive convergence trend of compared algorithms. The proposed DhoH tends to descend much faster than others and still maintains the optimization process long enough before reaching the convergence point.

3) STABILITY PERFORMANCE

To evaluate the stability aspect, we focus on the boxplot representation of values obtained after ten runs for some functions provided in Fig. 7. The boxes with larger margins and a longer rectangular shape mean that the variance of the corresponding method is high, which results in unstable output. These figures pinpoint that the most stable methods are DhoH and SMA since their results have low variances for most functions. Thus, we can confirm that the proposed method has decent stability.

C. PRACTICAL EXPERIMENT

The decentralization quality of block producers was evaluated and compared among the proposed DhoH algorithm and

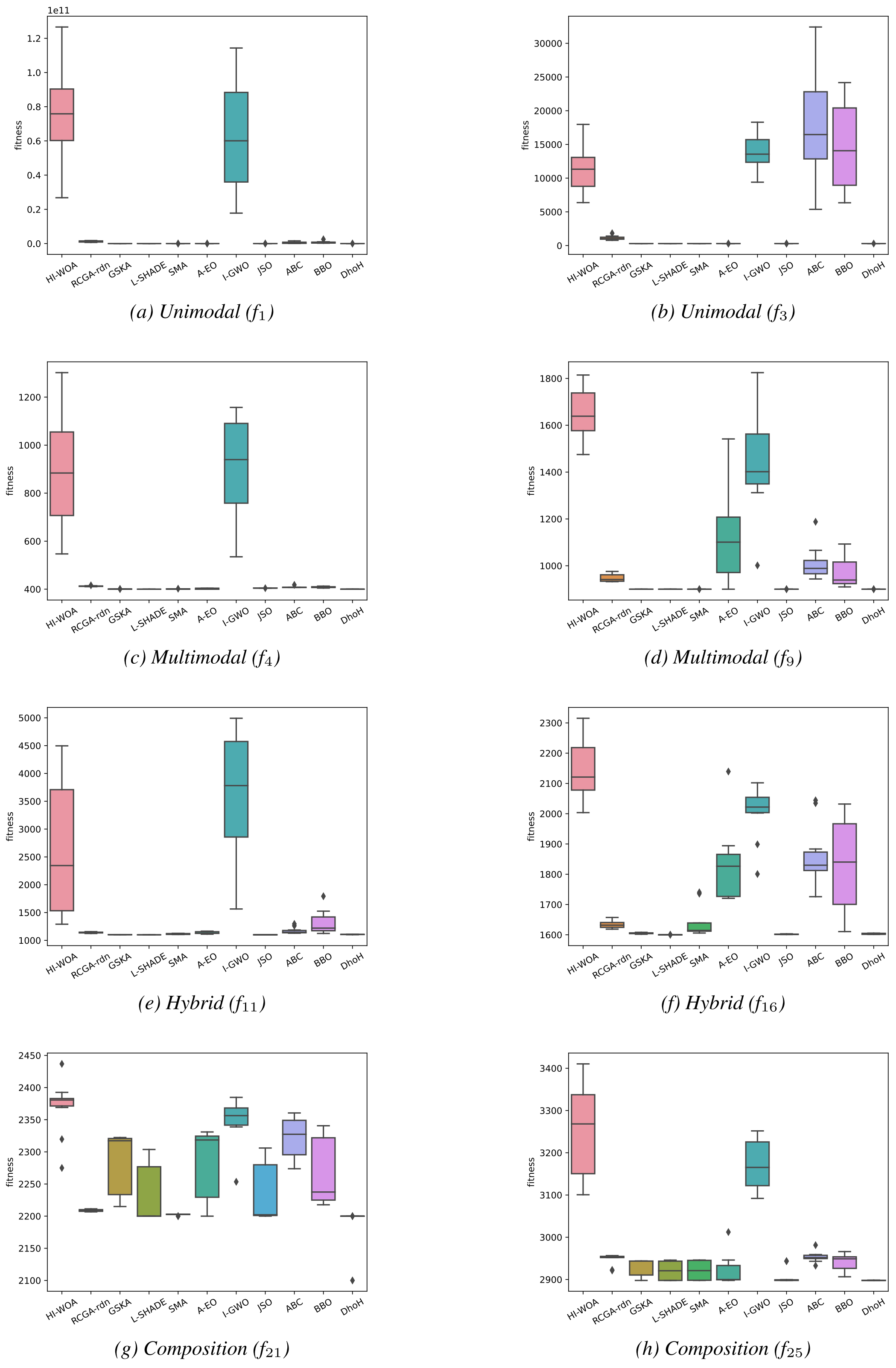


FIGURE 7. Visualization of the stability of some algorithms on benchmark functions.

six well-known algorithms. The mean value of the maximum decentralized level on ten runs is provided in Table. 5. As can be seen, the proposed algorithm gives approximate results of 0.254237, followed by L-SHADE and A-EO at 0.254181 and 0.254193. Almost all algorithms show that the decentralized quality is high and stable. DhoH is the stablest in most

TABLE 5. The decentralized level results of the DhoH algorithm versus others.

MPOC		
Algorithm	Decentralize Level	Standard error
HI-WOA	0.252010	3.20E-3
DhoH	**0.254237**	**5.55E-17**
GSKA	0.253824	9.46E-4
L-SHADE	0.254181	1.24E-4
SMA	0.253943	3.58E-3
A-EO	0.254194	6.60E-5

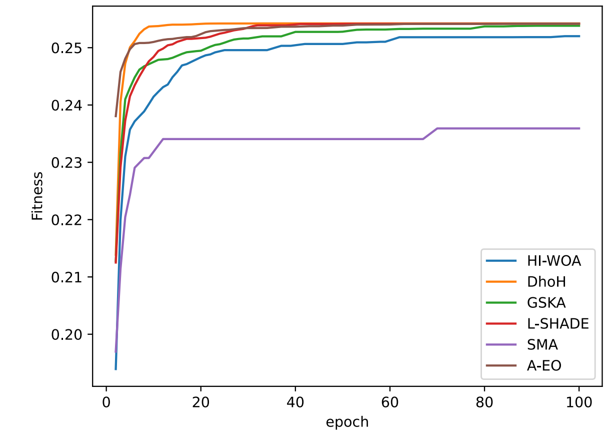


FIGURE 8. The convergence chart of DhoH and other algorithms in the practical test.

cases, with a standard error of approximately 5.55E-17. According to the convergence trend graph shown in Fig. 8, the convergence of DhoH is faster than other algorithms. The obtained experimental results are slightly similar to those in the theoretical experiment, showing that each algorithm's stability is promising. However, to get the best results on complex and multi-dimensional space, it is necessary to have an algorithm with good coverage and a suitable optimization method. We assume that DhoH is ideal for these problems, especially for the decentralized level optimization problem for the blockchain network.

V. CONCLUSION

A novel swarm-inspired optimization technique called the DhoH algorithm, inspired by the unique hunting behaviors and adapting skills in the new environment of dholes in nature, is proposed for dealing with different optimization tasks in this work. Both local and global searches of general metaheuristics have been deployed in DhoH. The pack-hunting process of dholes represents local search. Besides, global search is described as the pack reforming process.

To demonstrate the performance of DhoH, we set up two types of experiments, including both the theoretical and practical aspects. On the theoretical aspect, four sets of different benchmark functions are implemented to evaluate the performance results of DhoH versus the other seven state-of-the-art algorithms. The results show that DhoH can find the global optimum and is competitive with other algorithms. In terms of practicality, we use DhoH and the other four algorithms to optimize the decentralization of the MPoC consensus protocol in the blockchain network. The results demonstrate the applicability and potential of the proposed optimizer in practice.

The standard version of DhoH is simple but effective with a few parameters. There are several directions to further improve the algorithm in future works: (1) integrate some stochastic operators such as chaotic maps, opposition-based learning techniques, and deterministic operators. (2) equipped with some components from different algorithms to form a hybridized version. (3) design DhoH to solve binary or multi-objective optimization problems.

REFERENCES

[1] R. Abbassi, A. Abbassi, A. A. Heidari, and S. Mirjalili, "An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models," *Energy Convers. Manage.*, vol. 179, pp. 362–372, Jan. 2019.
[2] H. Faris, A. M. Al-Zoubi, A. A. Heidari, I. Aljarah, M. Mafarja, M. A. Hassonah, and H. Fujita, "An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks," *Inf. Fusion*, vol. 48, pp. 67–83, Aug. 2019.
[3] J. Nocedal and S. Wright, *Numerical Optimization*. Cham, Switzerland: Springer, 2006.
[4] G. Wu, "Across neighborhood search for numerical optimization," *Inf. Sci.*, vol. 329, pp. 597–618, Feb. 2016.
[5] Z. Beheshti and S. M. H. Shamsuddin, "A review of population-based meta-heuristic algorithms," *Int. J. Adv. Soft. Comput. Appl.*, vol. 5, no. 1, pp. 1–35, 2013.
[6] T. Nguyen, N. Tran, B. M. Nguyen, and G. Nguyen, "A resource usage prediction system using functional-link and genetic algorithm neural network for multivariate cloud metrics," in *Proc. IEEE 11th Conf. Service-Oriented Comput. Appl. (SOCA)*, Nov. 2018, pp. 49–56, doi: 10.1109/SOCA.2018.00014.
[7] T. Nguyen, B. M. Nguyen, and G. Nguyen, "Building resource auto-scaler with functional-link neural network and adaptive bacterial foraging optimization," in *Proc. Int. Conf. Theory Appl. Models Comput.*, 2019, pp. 501–517.
[8] A. N. Ahmed, T. Van Lam, N. D. Hung, N. Van Thieu, O. Kisi, and A. El-Shafie, "A comprehensive comparison of recent developed meta-heuristic algorithms for streamflow time series forecasting problem," *Appl. Soft Comput.*, vol. 105, Jul. 2021, Art. no. 107282.
[9] T. Nguyen, T. Nguyen, Q.-H. Vu, T. T. B. Huynh, and B. M. Nguyen, "Multi-objective sparrow search optimization for task scheduling in fog-cloud-blockchain systems," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Sep. 2021, pp. 450–455.
[10] E.-G. Talbi, *Metaheuristics: From Design to Implementation*, vol. 74. Hoboken, NJ, USA: Wiley, 2009.
[11] J. H. Holland, "Genetic algorithms," *Sci. Amer.*, vol. 267, no. 1, pp. 66–73, 1992.
[12] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS. Proc. 6th Int. Symp. Micro Mach. Hum. Sci.*, Oct. 1995, pp. 39–43.
[13] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic," in *Proc. Congr. Evol. Comput.*, vol. 2, Jul. 1999, pp. 1470–1477.
[14] H. A. Abbass, "MBO: Marriage in honey bees optimization—A haplometrosis polygynous swarming approach," in *Proc. Congr. Evol. Comput.*, vol. 1, May 2001, pp. 207–214.
[15] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Syst. Mag.*, vol. 22, no. 3, p. 52, Jun. 2002.
[16] X. Li, "A new intelligent optimization-artificial fish swarm algorithm," Ph.D. thesis, Zhejiang Univ., Zhejiang, China, 2003.
[17] D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, document TR06, 2005.
[18] D. Moldovan, V. Chifu, I. Salomie, and A. Slowik, "Cat swarm optimization," in *Swarm Intelligence Algorithms*. Cham, Switzerland: Cham, Switzerland: Springer, 2020, pp. 55–69.

[19] R. Martin and W. Stephen, "Termite: A swarm intelligent routing algorithm for mobilewireless ad-hoc networks," in *Studies in Computational Intelligence*. Cham, Switzerland: Springer, 2006, pp. 155–184.

[20] C. Yang, X. Tu, and J. Chen, "Algorithm of marriage in honey bees optimization based on the wolf pack search," in *Proc. Int. Conf. Intell. Pervasive Comput. (IPC)*, Oct. 2007, pp. 462–467.

[21] A. Mucherino, O. Seref, O. Seref, O. E. Kundakcioglu, and P. Pardalos, "Monkey search: A novel metaheuristic search for global optimization," in *AIP Conf. Proc.*, 2007, p. 162.

[22] X. Lu and Y. Zhou, "A novel global convergence algorithm: Bee collecting pollen algorithm," in *Proc. Int. Conf. Intell. Comput.*, 2008, p. 518.

[23] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, Dec. 2009, pp. 210–214.

[24] Y. Shiqin, J. Jianjun, and Y. Guangxing, "A dolphin partner optimization," in *Proc. WRI Global Congr. Intell. Syst.*, vol. 1, May 2009, pp. 124–128.

[25] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization* (NICSO). Cham, Switzerland: Springer, 2010, pp. 65–74.

[26] X.-S. Yang, "Firefly algorithm, Lévy flights and global optimization," in *Research and Development in Intelligent Systems*. Cham, Switzerland: Springer, 2010, pp. 209–218.

[27] Y. Marinakis and M. Marinaki, "Bumble bees mating optimization algorithm for the vehicle routing problem," in *Adaptation, Learning, and Optimization*. Cham, Switzerland: Springer, 2011, pp. 347–369.

[28] W.-T. Pan, "A new fruit fly optimization algorithm: Taking the financial distress model as an example," *Knowl.-Based Syst.*, vol. 26, pp. 69–74, Feb. 2012.

[29] A. H. Gandomi and A. H. Alavi, "Krill herd: A new bio-inspired optimization algorithm," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 17, no. 12, pp. 4831–4845, Dec. 2012.

[30] A. Askarzadeh and A. Rezazadeh, "A new heuristic optimization algorithm for modeling of proton exchange membrane fuel cell: Bird mating optimizer," *Int. J. Energy Res.*, vol. 37, no. 10, pp. 1196–1204, Aug. 2013.

[31] A. Kaveh and N. Farhoudi, "A new optimization method: Dolphin echolocation," *Adv. Eng. Softw.*, vol. 59, pp. 53–70, May 2013.

[32] X. Meng, Y. Liu, X. Gao, and H. Zhang, "A new bio-inspired algorithm: Chicken swarm optimization," in *Proc. Int. Conf. swarm Intell.*, 2014, pp. 86–94.

[33] H.-S. Wu and F.-M. Zhang, "Wolf pack algorithm for unconstrained global optimization," *Math. Problems Eng.*, vol. 2014, pp. 1–17, Jun. 2014.

[34] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Jun. 2014.

[35] J. C. Bansal, H. Sharma, S. S. Jadon, and M. Clerc, "Spider monkey optimization algorithm for numerical optimization," *Memetic Comput.*, vol. 6, no. 1, pp. 31–47, Mar. 2014.

[36] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.

[37] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 1995–2014, May 2015, doi: 10.1007/s00521-015-1923-y.

[38] O. Abedinia, N. Amjady, and A. Ghasemi, "A new metaheuristic algorithm based on shark smell optimization," *Complexity*, vol. 21, no. 5, pp. 97–116, May 2016.

[39] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Comput. Struct.*, vol. 169, pp. 1–12, Jun. 2016.

[40] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, Sep. 2016.

[41] X.-B. Meng, X. Z. Gao, L. Lu, Y. Liu, and H. Zhang, "A new bio-inspired optimisation algorithm: Bird swarm algorithm," *J. Experim. Theor. Artif. Intell.*, vol. 28, no. 4, pp. 673–687, Jul. 2016.

[42] M. Yazdani and F. Jolai, "Lion optimization algorithm (LOA): A nature-inspired metaheuristic algorithm," *J. Comput. Design Eng.*, vol. 3, no. 1, pp. 24–36, Jan. 2016.

[43] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: Theory and application," *Adv. Eng. Softw.*, vol. 105, pp. 30–47, Mar. 2017.

[44] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[45] G. Dhiman and V. Kumar, "Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications," *Adv. Eng. Softw.*, vol. 114, pp. 48–70, Dec. 2017.

[46] A. Kaveh and S. Mahjoubi, "Lion pride optimization algorithm: A meta-heuristic method for global optimization problems," *Scientia Iranica*, vol. 25, pp. 3113–3132, Aug. 2018.

[47] G. Dhiman and V. Kumar, "Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems," *Knowl.-Based Syst.*, vol. 165, pp. 169–196, Feb. 2019.

[48] M. Jain, V. Singh, and A. Rani, "A novel nature-inspired algorithm for optimization: Squirrel search algorithm," *Swarm Evol. Comput.*, vol. 44, pp. 148–175, Feb. 2019.

[49] H. Yapici and N. Cetinkaya, "A new meta-heuristic optimizer: Pathfinder algorithm," *Appl. Soft Comput.*, vol. 78, pp. 545–568, May 2019.

[50] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.

[51] S. Shadravan, H. R. Naji, and V. K. Bardsiri, "The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems," *Eng. Appl. Artif. Intell.*, vol. 80, pp. 20–34, Apr. 2019.

[52] R. Masadeh, A. A. Basel, and A. Sharieh, "Sea lion optimization algorithm," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 5, pp. 1–12, 2019.

[53] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[54] N. S. Jaddi and S. Abdullah, "Optimization of neural network using kidney-inspired algorithm with control of filtration rate and chaotic map for real-world rainfall forecasting," *Eng. Appl. Artif. Intell.*, vol. 67, pp. 246–259, Jan. 2018.

[55] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[56] B. M. Nguyen, T. Nguyen, T. Nguyen, and B.-L. Do, "MPoC—A metaheuristic proof of criteria consensus protocol for blockchain network," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2021, pp. 1–8, doi: 10.1109/ICBC51069.2021.9461113.

[57] P. S. Craig, P. Giraudoux, Z. H. Wang, and Q. Wang, "Echinococcosis transmission on the Tibetan Plateau," *Adv. Parasitology*, vol. 104, pp. 165–246, Jul. 2019.

[58] B. M. Nguyen, B. Hoang, T. Nguyen, and G. Nguyen, "NQSV-Net: A novel queuing search variant for global space search and workload modeling," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 1, pp. 27–46, Jan. 2021.

[59] O. Tarkhaneh and H. Shen, "Training of feedforward neural networks for data classification using hybrid particle swarm optimization, mantegna Lévy flight and neighborhood search," *Heliyon*, vol. 5, no. 4, Apr. 2019, Art. no. e01275.

[60] A. Pnsnh, M. Ali, J. Liang, and B. Qu, "Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization," Technol. Rep., 2017.

[61] N. Van Thieu and S. Mirjalili, "MEALPY: An open-source library for latest meta-heuristic algorithms in Python," *J. Syst. Archit.*, vol. 139, Jun. 2023, Art. no. 102871.

[62] C. Tang, W. Sun, W. Wu, and M. Xue, "A hybrid improved whale optimization algorithm," in *Proc. IEEE 15th Int. Conf. Control Autom. (ICCA)*, Jul. 2019, pp. 362–367.

[63] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Appl. Math. Comput.*, vol. 214, no. 1, pp. 108–132, Aug. 2009.

[64] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.

[65] Y. Song, F. Wang, and X. Chen, "An improved genetic algorithm for numerical function optimization," *Int. J. Speech Technol.*, vol. 49, no. 5, pp. 1880–1902, May 2019.

[66] A. W. Mohamed, A. A. Hadi, and A. K. Mohamed, "Gaining-sharing knowledge based algorithm for solving optimization problems: A novel nature-inspired algorithm," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 7, pp. 1501–1529, Jul. 2020.

[67] A. Wunnava, M. K. Naik, R. Panda, B. Jena, and A. Abraham, "A novel interdependence based multilevel thresholding technique using adaptive equilibrium optimizer," *Eng. Appl. Artif. Intell.*, vol. 94, Sep. 2020, Art. no. 103836.

[68] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1658–1665.
[69] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, p. 341, 1997.
[70] J. Brest, M. S. Maucec, and B. Boskovic, "Single objective real-parameter optimization: Algorithm jSO," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1311–1318.
[71] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: A new method for stochastic optimization," *Future Gener. Comput. Syst.*, vol. 111, pp. 300–323, Oct. 2020.
[72] M. H. Nadimi-Shahraki, S. Taghian, and S. Mirjalili, "An improved grey wolf optimizer for solving engineering problems," *Expert Syst. Appl.*, vol. 166, Mar. 2021, Art. no. 113917.
[73] N. Van Thieu and S. Mirjalili, "MEALPY: An open-source library for latest meta-heuristic algorithms in Python," *J. Syst. Archit.*, vol. 139, 2023, Art. no. 102871.
[74] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," White paper, 2008.
[75] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *Self-Published Pape*, vol. 19, no. 1, pp. 1–6, 2012.
[76] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (PoET)," in *Proc. Int. Symp. Stabil., Saf., Security Distrib. Syst. (SSS)*, 2017, pp. 282–297.

BINH MINH NGUYEN received the Dipl.-Ing. degree in computer-aided design from the Institute of Automation and Information Technologies, Tambov State Technical University, Russia, in 2008, and the Ph.D. degree in applied informatics from the Faculty of Informatics and Information Technology, Slovak University of Technology (STU) in Bratislava, Slovakia, in 2013. From 2008 to 2013, he worked as a Researcher with the Institute of Informatics, Slovak Academy of Sciences (IISAS), Slovakia. Currently, he is an Associate Professor with the School of Information and Communication Technology (SoICT), and the Director of the Research Center on Fintech, Hanoi University of Science and Technology, Vietnam. His research interests include distributed systems and data analytics for several application domains, including cloud/fog computing, blockchain, and financial technology.

THIEU NGUYEN received the B.Sc. and M.Sc. degrees from the School of Information and Communication Technology (SoICT), Hanoi University of Science and Technology (the ICT Talented Engineering Program). From 2018 to 2021, he was a Research Assistant with the High Performance Computing Center, HUST. He contributed as the first author to many papers in international conferences and journals with research topics focused on nature-inspired computing, machine learning, neural networks, and meta-heuristic algorithms in some specific domains, such as cloud computing and blockchain networks.

QUOC-HIEN VU is pursuing the degree in computer science with the School of Information and Communication and Technology (ICT), Hanoi University of Science and Technology. His research interests include algorithms for deep learning, reinforcement learning, meta-heuristic, and bioinformatics.

TRAN HUY HUNG received the bachelor's degree in computer science from the School of Information and Communication Technology (SoICT), Hanoi University of Science and Technology (HUST). His current research interests include multi-objective optimization, computational intelligence, and artificial intelligence.

TRAN HOANG HAI received the B.S. degree from Hanoi University of Science and Technology, Vietnam, the M.S. degree in computer engineering from Kyung Hee University, South Korea, in 2008, and the Ph.D. degree in computer science from the University of Rennes 1, France, in 2012. Since then, he has worked with INRIA Joint Alcatel-Lucent Bell Laboratory. He is currently a Lecturer with the School of Information and Communication Technology, Hanoi University of Science and Technology. His research interests include applied game theory and machine learning models to network security, routing, and resource allocation mechanisms in the next-generation internet. He has published several articles on those issues.

HUYNH THI THANH BINH is currently an Associate Professor and the Vice Dean of the School of Information and Communication Technology, Hanoi University of Science and Technology. She is the Head of the Modeling, Simulation and Optimization Laboratory (MSO). Her current research interests include computational intelligence, artificial intelligence, memetic computing, and evolutionary multitasking. She is a member of the IEEE Computational Intelligence Society-Women in Computational Intelligence Committee and an IEEE Asia Pacific Executive Committee Member. She has served as a Regular Reviewer and a Program Committee Member for numerous prestigious academic journals and conferences, such as *Applied Soft Computing*, *Memetic Computing*, IEEE Access, *Swarm and Evolutionary Computation*, IEEE Transactions on Evolutionary Computation. She was the IEEE Asia Pacific Student Activities Committee Chair, in 2019 and 2020. She is the Chair of the IEEE Computational Intelligence Society Vietnam Chapter. She is an Editor Board of *Engineering Applications of Artificial Intelligence* and *Journal of Computer Science and Cybernetics*. For more details please visit her homepage: https://users.soict.hust.edu.vn/binhht/.

VAN-DANG TRAN received the B.Sc. degree from the School of Information and Communication Technology, Hanoi University of Science and Technology (HUST), Vietnam, in 2017, and the Ph.D. degree in informatics from The Graduate University for Advanced Studies, Japan, in 2022. He is currently a Lecturer with the School of Information and Communication Technology, HUST. His research interests include databases, data processing and analysis, and software engineering for data management. Before joining HUST, in 2023, he was a Research Fellow of Japan Society for the Promotion of Science (DC2 & PD), from 2022 to 2023.