IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 5, 2020, accepted April 14, 2020, date of publication April 20, 2020, date of current version May 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2988717

Hybridization of Galactic Swarm and Evolution Whale Optimization for Global Search Problem

BINH MINH NGUYEN[1], TRUNG TRAN[1], THIEU NGUYEN[1], AND GIANG NGUYEN[2]
[1]School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi 100000, Vietnam
[2]Institute of Informatics, Slovak Academy of Sciences, 845 07 Bratislava, Slovakia

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

This work was supported by the Vietnamese MOET's Project (Researching and Applying Blockchain Technology to the Problem of Authenticating the Certificate Issuing Process in Vietnam) under Grant number B2020-BKA-14.

ABSTRACT The works presented in this paper addresses the robust population-based global optimization that is influenced by the simplicity and efficiency principles introduced in two new generation optimization algorithms. Galactic Swarm Optimization is inspired by the motion of stars, galaxies, and superclusters of galaxies under the influence of gravity. It acts well as a global controller of the whole optimization process by employing multiple flexible cycles of exploration and exploitation phases to find new, better solutions. However, the optimization process still suffers poverty in the exploitation phase, which is improved in this work by its hybridization with our evolution version of the Whale Optimization Algorithm. Concretely, the exploitation phase of Galactic Swarm Optimization is replaced by our Evolution Whale Optimization Algorithm to avoid early convergence. The Whale Optimization Algorithm mimics the unusual social behaviors and the hunting activities of humpback whales. However, it is not optimized for global optimization when the number of dimensions is increased. Hence its is evolved in our works by Levy-Flight trajectory for faster local search with adaptive step lengths and two-point crossover operator to reduce bias in the offspring creation procedure. The achieved results through extensive and careful experiments showed that our hybridization and evolution enhancements bring outstanding performance in terms of accuracy, convergence speed, and stability.

INDEX TERMS Global optimization, hybridization, evolution computation, galactic swarm optimization, whale optimization algorithm.

I. INTRODUCTION

Optimization is one of the most common and important issues in both research and development of science and technology. The real-world optimization problems often appear with high complexity and dimension. These problems are known as large-scale global optimization (or search) and cannot be solved by traditional optimization algorithms because of the high dimensions and complex interactions among variables. The large-scale global optimization problems exist extensively in many scientific research and engineering applications like distributed systems, large-scale machine learning, and neural networks optimization [1]–[3].

Meta-heuristic algorithms used to solve many current optimization problems in different domains such as system control [4], [5], machine design [6], Big Data [7], [8], engineering design problems [9]–[11], global optimization problems [12]–[14] and so forth. The algorithms could be classified roughly into two categories [15]:

- Single-solution-based algorithms randomly generate a certain solution and gradually improve the solution until the best results are obtained. The typical example is Simulated Annealing. The main drawback of this technique is that it may be trapped into a local optimum, which prevents to find out global optimum.
- Population-based algorithms generate a set of random solutions within a given search space. The typical examples are the Genetic Algorithm or Particle Swarm Optimization. These solutions are updated continuously until the best one is found out.

All meta-heuristic techniques operate the exploration and exploitation phases in the search space. At the same time, they also have to maintain the right balance between these two phases. The exploration investigates various promising areas in a search space while the exploitation finds optimal

The associate editor coordinating the review of this manuscript and approving it for publication was Roberto Pietrantuono.

solutions around them [16]–[18]. Therefore, there is a need to control these two phases effectively to achieve optimal solutions for an optimization problem.

The standard meta-heuristic algorithms for solving the large-scale global optimization problems often suffer the degradation of optimization ability when the search space dimension goes up. Thus, the performance of these algorithms deteriorates when tackling serious dimensional problems. There are two major reasons for the performance deterioration of these algorithms: (1) the increasing problem dimension size leads to its space complexity and characteristic alterations; (2) the search space exponentially increases with the problem size; so an optimization algorithm must be able to explore the entire search space efficiently, and this is not a trivial task.

Following this context, the scope of this work presented in this paper aims to challenge the dimension increase in global optimization using meta-heuristic solutions. The most notable feature of this work is the proposed optimizer with the name Hybridization of Galactic Swarm Optimization and Evolution Whale Optimization Algorithm (called HGEW). This evolution hybridization optimizer composes two population-based meta-heuristic algorithms:

- Galactic Swarm Optimization (GSO), and
- Our proposed evolution version of the Whale Optimization Algorithm (WOA). The proposed evolution version (called EWOA in short) improves the original one by Levy-Flight trajectory and two-point crossover operator.

HGEW was carefully tested as the whole with the set of CEC 2014 competition benchmark functions consisting of both unimodal, multi-modal, hybrid, and composition types, whose complexity characteristic is expanded following the increase in the number of dimensions. The experiments also compare the performance of HGEW with variants of GSO and WOA. The obtained outcomes showed that HGEW brings outstanding performance in terms of accuracy, convergence speed, and stability in solving the global optimization problems robustly. The proposed method introduced in this paper could be applied to optimization problems in general, like training neural network serving prediction tasks.

The structure of this paper is as follows. Section II presents a background and related work. It contains an overview of the optimization problem, large-scale global optimization, meta-heuristic solutions, as well as GSO and WOA in detail. The HGEW hybridization design composed of GSO and EWOA are presented in Section III. The experimental setup is described in Section IV. Section V brings experimental obtained results, comparisons and evaluations. Finally, Section VI concludes the study and defines our plans.

II. BACKGROUND AND RELATED WORK

A. OPTIMIZATION AND META-HEURISTICS

In general, an optimization problem can be solved using the exact or approximate approach. However, domain data usually have various and high characteristic dimensions, which lead to NP-hard problems. It is difficult to solve these problems by exact methods like fast steepest, sequential quadratic programming, conjugate gradient, and quasi-Newton methods. Approximate methods are promising to solve the above-mentioned issues by effectively finding feasible solutions. The approximate methods could be divided into two subclasses, namely heuristic and meta-heuristic algorithms.

Heuristic algorithms are often used for optimization problems based on several assumptions, such as knowledge inside a concrete domain [19]. Meta-heuristic algorithms are more domain-independent, and often operate based on probabilistic rules rather than deterministic ones. These algorithms are considered as a more general approach to solving almost all optimization problems with several advantages [20], [21], including: 1) High flexibility due to considering the optimization problems as black boxes; 2) Independence with gradient information; 3) Easy implementation for several fields; 4) Exploration capability enables to avoid local optima issue. As presented in Section I, meta-heuristic techniques are divided into single-solution-based and population-based algorithms. The works presented in this paper are in the population-based group.

B. POPULATION-BASED META-HEURISTIC ALGORITHMS

Population-based meta-heuristic algorithms often mimic the natural phenomena [22]–[24]. In this direction, these algorithms start the optimization process by generating a set (or population) of individuals. Each individual represents a candidate solution for the optimization problem. The population will be evolved iteratively by replacing the current population with newly generated ones using some often-stochastic operators [25], [26]. The optimization process is proceeded until satisfying a stopping criterion (e.g., the maximum number of iterations [27], [28]). With inspiration, population-based algorithms are categorized into: 1) *Evolutionary-based* group mimics biological evolutionary behaviors such as recombination, mutation, and selection; 2) *Physics-based* group is inspired by the physical laws; 3) *Human-based* group mimics certain human behaviors; 4) *Swarm-based* group mimics the social behaviors, e.g., decentralized and self-organized systems of organisms living in swarms, flocks, or herds.

Particle Swarm Optimization (PSO) [29] is a very first well-known swarm-based optimization. It is the premise of many other meta-heuristic algorithms proposed in recent years. In PSO, a swarm contains several candidate solutions (also known as particles), which coexist in the search space of the problem with D dimensions. The solution often cooperates and flies together to land on optimal personal positions. Over time, the best personal position (its own best position in the past) of each particle and the global best position (the current best position of the entire swarm) are recorded. The next position of a particle is updated based on the personal best (cognitive behavior) and the global best (social communication). PSO combines local search (through personal best) with global search (through global best) to

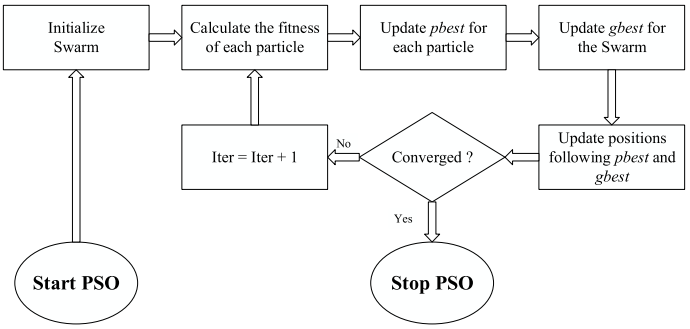


FIGURE 1. Particle swarm optimization operation workflow.

balance exploitation and exploration processes, as illustrated in Figure 1 with its operation workflow.

GSO and WOA, which are in the center of interest of our works belonging to the swarm-based group. The common exciting feature of both algorithms is that they offer the simplicity and efficiency principles presented in the following subsections.

C. GALACTIC SWARM OPTIMIZATION (GSO)

GSO [30] is inspired by the behaviors of stars in galaxies, and of galaxies in superclusters in the cosmos. It takes advantage of the original PSO by using multiple flexible cycles of exploration and exploitation phases to find new, better solutions. Due to the simple implementation and efficiency, GSO has been applied to several real-life problems [31]–[35] as well as in searching global optimization [36], [37].

In the original GSO, under the influence of gravity, stars in a galaxy are attracted to another star, which has greater gravity. From these ideas, the movement of stars inside a galaxy as well as the movement of galaxies are emulated in the GSO algorithm by the following rules:

- Individuals in each galaxy are attracted to bigger ones (i.e., better solutions) in their galaxy. The attraction process is performed by using the PSO algorithm.
- Global bests of all galaxies are chosen to treat as a super-warm. Here, the PSO algorithm is used again to represent the movement of particles in the super-swarm.

GSO's pseudocode is presented in Algorithm 1, and the variable explanation is shown in detail by Table 1.

1) INITIALIZING STAGE

The main control parameters of whole GSO algorithm consists of its parameters used in each phase (Table 1). Swarm (population) includes M partitions called sub-swarms X_i containing N elements ($x_j^{(i)} \in \mathbb{R}^D$). Each element is created randomly within the search space $[x_{min}, x_{max}]^D$.

$$X = \{X_i | i = 1, 2, \ldots, M\} \tag{1}$$

Algorithm 1 Galactic Swarm Optimization (GSO)

Initialize $x_j^{(i)}, v_j^{(i)}, p_j^{(i)}, g_j^{(i)}$ within $[x_{min}, x_{max}]^D$ randomly.
Initialize $v^{(i)}, p^{(i)}, g$ within $[x_{min}, x_{max}]^D$ randomly.
for $Iter = 0$ to $Iter_{max}$ **do**
  Begin PSO: Level 1
  for $i = 1$ to M **do**
    for $k = 0$ to L_1 **do**
      for $j \leftarrow 1$ to N **do**
        $v_j^{(i)} \leftarrow \omega_1 v_j^{(i)} + c_1 r_1 (p_j^{(i)} - x_j^{(i)}) + c_2 r_2 (g^{(i)} - x_j^{(i)})$
        $x_j^{(i)} \leftarrow x_j^{(i)} + v_j^{(i)}$
        if $f(x_j^{(i)}) < f(p_j^{(i)})$ **then**
          $p_j^{(i)} \leftarrow x_j^{(i)}$
          if $f(p_j^{(i)}) < f(g^{(i)})$ **then**
            $g^{(i)} \leftarrow p_j^{(i)}$
          end if
        end if
      end for
    end for
  end for
  Begin PSO: Level 2
  Initialize Swarm $y^{(i)} = g^{(i)} : i = 1, 2, \ldots, M$;
  for $k = 0$ to $L2$ **do**
    for $i = 1$ to M **do**
      $v^{(i)} \leftarrow \omega_2 v^{(i)} + c_3 r_3 (p^{(i)} - y^{(i)}) + c_4 r_4 (g - y^{(i)})$
      $y^{(i)} \leftarrow y^{(i)} + v^{(i)}$
      if $f(y^{(i)}) < f(p^{(i)})$ **then**
        $p^{(i)} \leftarrow y^{(i)}$
        if $f(p^{(i)}) < f(g)$ **then**
          $g \leftarrow p^{(i)}$;
        end if
      end if
    end for
  end for
end for
Results: $g, f(g)$

TABLE 1. Galactic swarm optimization parameters.

Symbol	Description
D	Dimension of a solution
f	Cost (objective) function
X	Population consists of all particles $x_j^{(i)}$ in the swarm
X_i	Subset of population belonging to subswarm i
$x_j^{(i)}$	Position of particle j in subswarm i
$v_j^{(i)}$	Velocity of particle $x_j^{(i)}$
$p_j^{(i)}$	Personal best of particle $x_j^{(i)}$
$g^{(i)}$	Global best of subswarm i
$y^{(i)}$	Position of particle i in superswarm (current global best of subswarm i)
$v^{(i)}$	Velocity of $y^{(i)}$
g	Global best solution of superswarm
$Iter_{max}$	The maximum number of iterations
M	Number of subswarms
N	Size of subswarm i
L_i	Number of iterations in phase i
c_1, c_2, c_3, c_4	Acceleration coefficient of PSO used in 2 phases
r_i	Uniform distributed random number in $[-1, 1]$
w_i	Inertial weight

$$X_i = \left\{X_j^{(i)} | j = 1, 2, \ldots, N\right\} \tag{2}$$

$$X_i \cap X_j = \varnothing \quad \forall i \neq j \tag{3}$$

2) UPDATING STAGE (EXPLORATION AND EXPLOITATION PHASE)

This stage covers two phases as follows:

- *Exploration of sub-swarms.* PSO algorithm is run for each sub-swarm. Since swarm X is initially divided into M groups, PSO will run M times independently with global bests $g^{(i)}$ tied to each sub-swarm. $g^{(i)}$ will be updated if any particles in the sub-swarm have personal best $p_j^{(i)}$, which is a better solution than $g^{(i)}$, $f(p_j^{(i)}) < f(g^{(i)})$. Each sub-swarm independently explores its best solution in its search space. This task is initialized by calculating velocity $v_j^{(i)}$ and position $x_j^{(i)}$ of particles.
- *Exploitation in super-swarm.* All global bests from M sub-swarms presented in the first phase above are gathered to form the super-swarm. In other words, a new super-swarm Y is created by collecting M global bests of each sub-swarm X_i.

$$Y = \left\{y^{(i)} | y^{(i)} = g^{(i)}, \forall i = 1, 2, \ldots, M\right\} \tag{4}$$

In this way, velocity $v^{(i)}$ and position $y^{(i)}$ are updated using PSO algorithm one again (the first use is in the first phase presented above). Thus, the super-warm utilizes the best solutions that are computed by sub-swarms in the previous phase, and then it initializes the population for the second phase of GSO. The population made by the best solutions from the first phase helps the PSO algorithm in the second phase to start at a better beginning point. Based on that, GSO achieves faster, accurate convergence.

However, GSO faces the problem of being trapped in local optima and slow convergence due to the poverty of exploitation and exploration [38]. Concretely, although PSO in the first phase of GSO could push initialized particles to move a giant step to the global minimum, the exploitation capability of PSO in the second phase is not good enough to escape the local minimums and to prevent premature convergence, which was indicated in [39].

D. WHALE OPTIMIZATION ALGORITHM (WOA)

WOA recently emerged as a beneficial algorithm, which is applied in many practical problems such as forecasting water resources demand [40], flow shop scheduling problem [41], opinion leader detection in online social network [42], medical diagnosis [43]. However, it was first proposed in [44], which mimicked the unusual social behaviors and the engaging hunting activities of humpback whales. Hunting prey operation of humpback whales is called bubble-net feeding, which is often done in groups. The group size can range from two or three and up to sixty whales participating at once. After the bubble net is executed, it is kept shrinking until the food is eaten [45].

In the beginning, due to the random initialization of the whale population, the position of prey is not known. WOA assumes that a solution with the best fitness will be the prey, and it is reset at the beginning of each iteration. After the target is recognized, other whales will try to update their position around the prey. The hunting process executed by humpback whales is represented perfectly by three operations as follows:

1) *Shrinking encircling mechanism* enables the reduction of the encircling magnitude to update position landed in the search space between the original solution position and the best one at the current iteration.
2) *Spiral updating position* is based on the way, or humpback whales approach their prey. A spiral equation is created to emulate the helix-shape movement of humpback whales within the space between the position of whale and prey.
3) *Search for prey* represents the exploration process while whales find prey randomly that enables WOA to escape the local minimum easily.

WOA, with its operations, are thoroughly discussed in [44]. We summarize the mechanisms of WOA by Algorithm 2. Although the original WOA with the encircling mechanism and the spiral path is good at exploring the search space, it outperforms other natural inspired algorithms from the perspective of simplicity and efficiency, WOA still requires a specific improvement for tackling the large-scale global optimization when it gets stuck into local optima and degrades accuracy [46].

E. MOTIVATION AND CONTRIBUTION OF THE WORK

The work presented in this paper addresses the global optimization problem. In our research direction, this optimizer could be used to find suitable parameters for the neural network in workload prediction and auto-scaling problems [22], [23], [47] of distributed systems like cloud computing and blockchain networks [48], [49].

Algorithm 2 Whale Optimization Algorithm (WOA)

Initialize the *population* $= \{X_i | i = 1, 2, \ldots, N\}$ randomly
Calculate fitness of each *solution* (whale)
$X_* \leftarrow$ the best solution
for $Iter = 0 \rightarrow Iter_{max}$ **do**
  for *solution* in *population* **do**
    Update a, A, C, l, p (Section II-D)
    if $p < 0.5$ **then**
      if $|A| < 1$ **then**
        Shrinking encircling mechanism
      else
        Search for prey process
      end if
    else
      Spiral updating position
    end if
  end for
  Evaluate population: if a feature of a *solution* goes beyond its boundary, set it the value of the boundary
  Recompute the fitness of all *solutions*
  Update X_* if a better *solution* is found.
end for
Results: $X_*, f(X_*)$

Our proposed approach is inspired by the simplicity and efficiency offered in GSO and WOA swarm-based algorithms. GSO acts well as a global controller of the whole optimization process, but it suffers early convergence in exploitation because of the limitations of the PSO algorithm used in the second phase of GSO (i.e., easily being trapped in a local minimum, especially in complex multi-peak search problems as shown in [50] and [38]. On the other hand, WOA possesses a decent exploitation capability as compared with other well-known nature-inspired algorithms (e.g., GA and PSO), as shown in [44]. In order to accelerate convergence speed as well as retain the diversity of the population in WOA, several variants of this algorithm are introduced in [51], which used chaotic technique, and in [52] used Levy-Flight trajectory technique. The results have shown in [52] that WOA variants outperform many algorithms such as Moth Flame Optimizer, Bat Algorithm, Artificial Bee Colony, and the origin WOA. However, the exploitation and exploration capabilities of WOA variants still need to be enhanced since they still face the problem of premature convergence when tackling problems having increasing dimensions. Based on those motivations, the main contributions of our works include:

1) Proposing an evolution hybridization between GSO and EWOA as a new optimizer to avoid early convergence in the exploitation phase while keeping advantages of the original GSO as the whole as well as its exploration power. The new evolution hybridization is called HGEW in short, which stands for the Hybridization of Galactic Swarm Optimization and Evolution Whale Optimization Algorithm.
2) Proposing an evolution version of WOA based on Levy-Flight trajectory and two-point crossover operator. The aim is to provide faster local search with adaptive step lengths in the HGEW exploitation phase and to reduce bias in the offspring creation procedure. The new evolution version of WOA is called EWOA in short, which stands for Evolution Whale Optimization Algorithm (EWOA).
3) Providing thorough evaluation of HGEW functionalities as the whole through extensive experiments with different high-dimensional benchmark functions of two types of unimodal and multi-modal functions.
4) Providing thorough validation HGEW functionalities in comparison with baseline optimizers such as GSO and WOA-based with Levy-Flight trajectory. HGEW is also carefully compared with other hybridization variances such as GSO with WOA and GSO with Levy-Flight WOA. The gained results prove the HGEW contribution values in the aspects: more stability and robustness in comparison with existing ones.

III. EVOLUTION HYBRIDIZATION OF GSO AND EWOA

As above presented in Section II-E, our proposed evolution hybridization HGEW is built as GSO hybridization with our evolved EWOA. HGEW acts as the global controller of the whole optimization process with exploration and exploitation phases. It replaces the exploitation phase of GSO by our proposed EWOA as follows.

A. EVOLUTION WHALE OPTIMIZATION ALGORITHM (EWOA)

The original WOA (Section II-D) can easily tackle problems of low-dimensional optimization. However, when applying WOA to global optimization problems with a higher dimension, it often trappers in local optima because the diversity population is decreased rapidly. To improve the exploration, convergence speed, and local minimum avoidance, several variants of WOA were introduced [46], [52]–[56]. The Levy-Flight trajectory [52] is one of the popular techniques used to improve the performance of meta-heuristics algorithms such as in Firefly [57], Krill Herd [58], Grey Wolf Optimization [25], [59], [60], Butterfly [61], and Harris Hawks Optimization [62].

In our work, WOA is evolved as EWOA in evolution way with enhanced searchability based on two ideas:

1) Using *Levy-Flight trajectory* to escape the local optima and accelerate the convergence speed,
2) Using *Crossover operator* to create more potential random offspring candidates in an evolution way than in [44].

1) LEVY-FLIGHT

The Levy-Flight [52] is a random process with step length chosen from Levy distribution. In WOA, Levy-Flight could help to maximize the diversification of newly created

Algorithm 3 Evolution Whale Optimization (EWOA)

Initialize the *population* $= \{X_i | i = 1, 2, \ldots, N\}$ randomly
Calculate fitness of each *solution* (whale)
X_* ← the best *solution*
for *Iter* $= 0$ to $Iter_{max}$ **do**
  for *solution* in *population* **do**
    Update a, A, C, l, p_1, p_2 (Section II-D)
    if $p_1 < 0.5$ **then**
      if $|A| < 1$ **then**
        Shrinking encircling mechanism based on *Levy-Flight*
      else
        if $p_2 < 0.4$ **then**
          Updating by *Crossover* operator
        else
          Search for prey (exploration phase)
        end if
      end if
    else
      Spiral updating position
    end if
  end for
  Evaluate population: if a feature of a *solution* goes beyond its boundary, set it the value of the boundary
  Recompute the fitness of all *solutions*
  Update X_* if a better *solution* is found
end for
Results: $X_*, f(X_*)$

solutions, and allows the algorithm to exploit more efficiently in the search space. Additionally, it also avoids the local minimum issue and increases the acceleration of convergence [46], [52]. The technique also supports to obtain the balance between exploration and exploitation phases. WOA thus can jump out easily when it gets stuck at a local minimum. The mathematical expression of the technique is as follows:

$$X^{t+1} = (X^t + \mu sign[rand - 0.5]) \oplus Levy \tag{5}$$

where

X^t	indicates the current position;
μ	is a random number chosen by uniform distribution;
rand	is random number in [0, 1];
$\oplus$	stands for entry-wise multiplication;
sign	$sign[rand - 0.5]$ has only one of three values -1, 1, or 0.

For *Levy* part, the law vision of the Levy distribution is:

$$L(s) \sim |s|^{-1-\beta} \quad with\ 0 < \beta \leq 2 \tag{6}$$

where β is an index, s is the step length of the Levy-Flight, which is calculated following Mantegna's algorithm (Equation 7).

$$s = \frac{\mu}{|v|^{1/\beta}} \tag{7}$$

where μ and v are chosen from normal distributions

$$\mu \sim N(0, \sigma_\mu^2) \quad \text{and} \tag{8}$$

$$v \sim N(0, \sigma_v^2) \tag{9}$$

$$\sigma_\mu = \left[\frac{\Gamma(1+\beta).\sin(\pi.\beta/2)}{\Gamma((1+\beta)/2).\beta.2^{(\beta-1)/2}}\right]^{1/\beta} \tag{10}$$

$$\sigma_v = 1 \tag{11}$$

A step size avoiding Levy-Flight jumping out the search space is defined by Equation 12.

$$Levy = random(size(D)) \oplus L(\beta) \tag{12}$$

$$L(\beta) \sim \frac{\mu}{|v|^{1/\beta}}(X_i - X_*) \tag{13}$$

where D is the dimension of search space, X_i and X^* are the current solutions and best solutions, respectively.

Levy-Flight models two movements of whale based on the infinite variance of Levy distribution: while the long-distance movement is used to enhance the ability of exploration, the short-distance is performed to increase the exploitation capability. In our EWOA, the *Shrinking encircling mechanism* is eliminated, and Levy-Flight is employed to replace it in order to explore and exploit in the search space more effectively. The mathematical model of the exploitation phase of EWOA thus is modified and represented as follows:

$$X^{t+1} = \begin{cases} (X^t + \mu sign[rand - 0.5]) \oplus Levy, & \text{if } p \geq 0.5 \\ D'.e^{bl}.\cos(2\pi l) + X_*^t, & \text{otherwise} \end{cases} \tag{14}$$

where $\mu = 1/sqrt(t)$ with t is the current iteration and $sqrt()$ stands for square root operation. *rand* is a number in range [0, 1], so that $sign[rand - 0.5]$ presents a sign function with three values $-1, 0, 1$, which makes the process more random (Algorithm 3).

2) CROSSOVER OPERATOR

As described above, the problems of the original WOA biodiversity reduction in global optimization need to be tackled too. In our work, a new operation inspired by the behaviors of humpback whales when they migrate to tropical water and give birth in mating seasons is proposed. In evolutionary-based meta-heuristics such as Genetic Algorithm [63], [64], or Differential Evolution [65], crossover operator plays a role as increasing the diversity of population. Therefore, to keep the population diversity in WOA throughout the iteration, we apply the operator to WOA. Thus, the operator uses the current best solution X_*, and a random solution in the current population Y is chosen as parents. Then they generate an offspring Z (new solution) following two-point crossover rule (Figure 2) with the following workflow:

- Firstly, two numbers i and j (two crossover points) are chosen randomly, ranging from 0 to dimension D. The

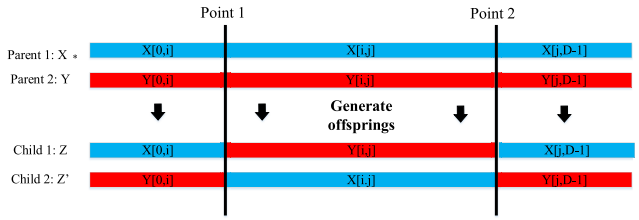


FIGURE 2. Two-point crossover operator.

choices of i and j must satisfy two main conditions: $i < j$ and $j - i = cr.D$ with cr is the operator crossover rate.

- Secondly, i and j divide a potential solution into two parts. In this way, the best solution X_*, for example, is divided into $X_*[i, j]$ part and the rest. The segments feature of solutions then is exchanged and creates two new offspring. One of them will be chosen as a new whale in the WOA model.

The mathematical process of *Crossover* operator is presented as follows:

$$Z[0, i] = Y[0, i] \tag{15}$$

$$Z[i, j] = X_*[i, j] \tag{16}$$

$$Z[j, D-1] = Y[j, D-1] \tag{17}$$

Unlike single-point Crossover, which faces a bias problem when the chosen point is near the end of features, a two-point crossover still works effectively in that case. Apart from this, the best solution X_* plays an important guiding role in this operation, which helps WOA explore a new potential solution as an offspring. Additionally, the randomly chosen solution Y keeps maintaining the diversity of population and enhancing exploitation. As *Shrinking encircling mechanism* was improved by Levy-Flight, and *Spiral updating position* was proved to perform a decent exploitation task in [44], we employ *Crossover* operator to improve the exploration phase of original WOA. However, in the crossover mechanism, if the operator is always used, population diversity will decrease dramatically. Hence, the operator is controlled through p_2 hyper-parameter in Algorithm 3.

B. HGEW: A HYBRIDIZATION OF GSO AND EWOA

As presented in Section II-C, GSO takes the advantages of the original PSO algorithm by using flexible multiple cycles of exploration and exploitation phases to escape local minimums and land to new, better solutions. However, using PSO still causes mediocre convergence issue [38]. Specifically, although PSO in the first phase of GSO can help initialized particles move a giant step to the global minimum, PSO in the second phase would not be able to exploit enough to escape the local minimums and to prevent premature convergence due to the restrictions of original PSO algorithm [66].

According to [44], [46], [52], as compared with PSO, WOA, and other its variants could bring better results with their exploration and exploitation. For that reason, in our HGEW evolution hybridization design, PSO at the second phase of the original GSO is replaced by EWOA. The HGEW workflow is presented in Algorithm 4.

Algorithm 4 HGEW Hybridization of GSO and EWOA

Initialize $x_j^{(i)}$, $v_j^{(i)}$, $p_j^{(i)}$, $g_j^{(i)}$, within $[x_{min}, x_{max}]^D$ randomly.
Initialize $v^{(i)}, p^{(i)}, g$ within $[x_{min}, x_{max}]^D$ randomly.
for $Iter = 0$ to $Iter_{max}$ **do**
  Begin PSO: Level 1
  for $i = 1$ to M **do**
    for $k = 0$ to L_1 **do**
      for $j \leftarrow 1$ to N **do**
        $v_j^{(i)} \leftarrow \omega_1 v_j^{(i)} + c_1 r_1 (p_j^{(i)} - x_j^{(i)}) + c_2 r_2 (g^{(i)} - x_j^{(i)})$
        $x_j^{(i)} \leftarrow x_j^{(i)} + v_j^{(i)}$
        if $f(x_j^{(i)}) < f(p_j^{(i)})$ **then**
          $p_j^{(i)} \leftarrow x_j^{(i)}$
          if $f(p_j^{(i)}) < f(g^{(i)})$ **then**
            $g^{(i)} \leftarrow p_j^{(i)}$
          end if
        end if
      end for
    end for
  end for
  Begin EWOA: Level 2
  Initialize Swarm $y^{(i)} = g^{(i)} : i = 1, 2, \ldots, M$
  for $k \leftarrow 0$ to $L2$ **do**
    for $i \leftarrow 1$ to M **do**
      Update a, A, C, l, p_1, p_2 (Section II-D)
      if $p_1 < 0.5$ **then**
        if $|A| < 1$ **then**
          Shrinking encircling mechanism based on *Levy-Flight*
        else
          if $p_2 < 0.4$ **then**
            Updating by *Crossover* operator
          else
            Search for prey (exploration phase)
          end if
        end if
      else
        Spiral updating position
      end if
      Evaluate *population*: Evaluate population: if a feature of a solution goes beyond its boundary, set it the value of the boundary
      Recompute the fitness of all *solutions*
      Update g if a better *solution* is found.
    end for
  end for
end for
Results: $g, f(g)$

TABLE 2. Benchmark functions (The implementation is available at [69]).

Type		Function name	Optimal value
Unimodal Functions	C1	Rotated High Conditioned Elliptic Function	100
	C2	Rotated Bent Cigar Function	200
	C3	Rotated Discus Function	300
Multimodal Functions	C4	Shifted and Rotated Rosenbrock's Function	400
	C5	Shifted and Rotated Ackley's Function	500
	C6	Shifted and Rotated Weierstrass Function	600
	C7	Shifted and Rotated Griewank's Function	700
	C8	Shifted Rastrigin's Function	800
	C9	Shifted and Rotated Rastrigin's Function	900
	C10	Shifted Schwefel's Function	1000
	C11	Shifted and Rotated Schwefel's Function	1100
	C12	Shifted and Rotated Katsuura Function	1200
	C13	Shifted and Rotated HappyCat Function	1300
	C14	Shifted and Rotated HGBat Function	1400
	C15	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	1500
	C16	Shifted and Rotated Expanded Scaffer's F6 Function	1600
Hybrid Functions	C17	Hybrid Function 1 (N=3)	1700
	C18	Hybrid Function 2 (N=3)	1800
	C19	Hybrid Function 3 (N=4)	1900
	C20	Hybrid Function 4 (N=4)	2000
	C21	Hybrid Function 5 (N=5)	2100
	C22	Hybrid Function 6 (N=5)	2200
Composition Functions	C23	Composition Function 1 (N=5)	2300
	C24	Composition Function 2 (N=3)	2400
	C25	Composition Function 3 (N=3)	2500
	C26	Composition Function 4 (N=5)	2600
	C27	Composition Function 5 (N=5)	2700
	C28	Composition Function 6 (N=5)	2800
	C29	Composition Function 7 (N=3)	2900
	C30	Composition Function 8 (N=3)	3000
Search Range: $[-100, 100]^D$			

- In the first phase, we retain PSO for exploration. PSO in the first phase thus works as an initializer, which decreases the search space and creates a better-initialized population used in the second phase.
- After that, EWOA in the second phase takes the advantages by using the population generated from the first phase to move faster to the global minimum.
  - Considering at the beginning of each iteration, the population in the second phase is always created again, but sub-swarms in the first phase continue exactly at the location where they left in the previous iteration. This process always allows EWOA to collect better individuals from subswarms throughout iterations when it starts to run.
  - On the other hand, the global best position in the second phase is recorded at the end in each iteration, and then it is employed in the next iteration, so that leads to a faster move and convergence in EWOA.

TABLE 3. Parameters used in GSO-based algorithms.

Dimension	M	N	L_1	L_2	$Iter_{max}$	c_i	FE
20	15	10	10	300	50	2.05	300000
50	15	10	10	300	50	2.05	300000
100	15	10	10	300	50	2.05	300000

TABLE 4. Parameters used in LWOA algorithm.

Dimension	P_{size}	$Iter_{max}$	FE
20	150	2000	300000
50	150	2000	300000
100	150	2000	300000

The source code of the proposed Hybridization of Galactic Swarm Optimization and Evolution Whale Optimization Algorithm (HGEW) is publicly available in [67] as Open Source under Apache 2.0 license.

TABLE 5. Results and comparisons of different algorithms on unimodal and multi-modal benchmark functions with 20D.

Function		GSO	GSO-LWOA	GSO-WOA	LWOA	HGEW
C1	mean	**100**	100	101	100	100
	std	**1.17E-04**	1.76E-01	1.21E+00	1.56E-01	1.75E-01
	rank	1	4	5	2	3
C2	mean	**200**	201	210	204	200
	std	**5.58E-05**	1.24E+00	1.80E+01	6.04E+00	8.15E-01
	rank	1	3	5	4	2
C3	mean	300	300	301	300	**300**
	std	7.12E-02	1.99E-01	3.60E-01	6.47E-01	**1.62E-07**
	rank	2	3	4	5	1
C4	mean	400	400	400	400	**400**
	std	8.29E-03	2.63E-07	9.31E-07	1.42E-07	**4.36E-08**
	rank	5	3	4	2	1
C5	mean	518	500	500	500	**500**
	std	5.10E+00	4.72E-04	2.22E-03	1.06E-03	**2.76E-04**
	rank	5	2	4	3	1
C6	mean	**600**	600	600	600	600
	std	**4.84E-13**	6.24E-09	3.70E-08	7.40E-09	1.76E-09
	rank	1	3	5	4	2
C7	mean	700	700	700	**700**	700
	std	1.39E-01	1.26E-05	1.27E-02	**1.04E-05**	2.16E-03
	rank	5	2	4	1	3
C8	mean	808	800	**800**	800	800
	std	2.70E+00	1.54E-06	**8.21E-06**	2.43E-06	4.81E-07
	rank	5	2	1	3	4
C9	mean	910	900	900	900	**900**
	std	2.69E+00	6.71E-07	1.12E-05	2.14E-06	**1.82E-07**
	rank	5	2	4	3	1
C10	mean	1470	1000	1000	1000	**1000**
	std	2.59E+02	2.86E-04	4.74E-04	3.29E-05	**2.18E-05**
	rank	5	3	4	2	1
C11	mean	1570	1100	**1100**	1100	1100
	std	3.33E+02	2.55E-05	**1.08E-01**	3.82E-05	6.93E-03
	rank	5	1	4	2	3
C12	mean	1200	1200	1200	1200	**1200**
	std	1.24E-01	1.46E-03	4.82E-03	3.95E-04	**1.52E-04**
	rank	5	3	4	2	1
C13	mean	1300	1300	1300	1300	**1300**
	std	1.64E-01	8.04E-02	1.35E-01	1.07E-01	**6.04E-02**
	rank	5	2	4	3	1
C14	mean	1400	1400	1400	1400	**1400**
	std	1.60E-01	8.73E-02	6.98E-02	1.00E-01	**6.19E-02**
	rank	5	3	2	4	1
C15	mean	1500	**1500**	1500	1500	1500
	std	7.81E-01	**2.16E-13**	3.01E-13	3.60E-13	2.44E-13
	rank	5	1	3	4	2
C16	mean	1600	1600	1600	1600	**1600**
	std	1.85E-01	1.76E-06	6.17E-06	2.02E-06	**1.05E-06**
	rank	5	2	4	3	1
	average rank	4	2.44	3.80	2.94	**1.82**
	overall rank	5	2	4	3	**1**

IV. EXPERIMENTS

Our HGEW evolution hybridization between GSO and EWOA is evaluated as the whole optimizer using different scalable benchmark functions. The following optimizers are used to compare the effectiveness with HGEW:

- GSO [30] original one with two PSO phases;
- LWOA as the improved version of WOA based on Levy-Flight trajectory [52];
- GSO-WOA as the hybridization with PSO [29] in the first phase and WOA [44] in the second phase;
- GSO-LWOA as the hybridization with PSO [29] the first phase and LWOA [52] in the second phase.

Several well-known optimization algorithms such as GA, PSO, and their variances, as well as Artificial Bee Colony and Ant Colony Optimization, are not considered in our experiments because they were compared with the improved version of WOA based on Levy-Flight trajectory (LWOA) [52].

Except for LWOA, the work [68] presents a systematic and meta-analysis survey of WOA usage with its modifi-

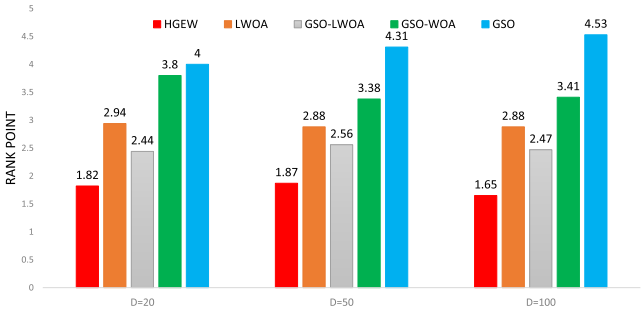


FIGURE 3. Average rankings of each algorithm for unimodal and multimodal functions.

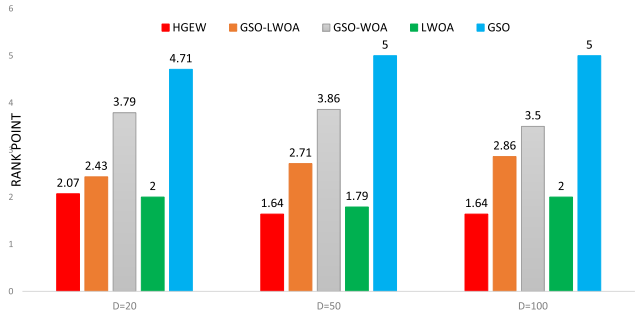


FIGURE 4. Average rankings of each algorithm for hybrid and composition functions.

cations such as adaptive, self-adaptive, improved, chaotic, modified, and memetic versions in different areas. From all of these modifications, LWOA has shown better performance compared to other variances in various domains. This is the reason to compare our proposed HGEW with LWOA also in combinations with GSO, i.e., LWOA, WOA-GSO, and LWOA-GSO in our experiments.

Thus, WOA and LWOA demonstrated its advantages over those above techniques. The demonstration is also the reason to explain why we used WOA as well as LWOA with GSO in our experiments to evaluate our proposed HGEW.

A. BENCHMARK FUNCTIONS

The work presented in this paper aims to address the global optimization problem, which stems from optimizing neural networks for workload prediction tasks. The set of 30 functions on Real Parameter Single Objective Optimization (incorporates expensive function optimization) of the CEC 2014 competition [70], [71] as benchmark functions for testing HGEW's performance in our experiments. Here is suitable to mention that many real-world problems belong to multi-objective optimization, which will be in the scope of our future works. These 30 benchmark functions are presented in Table 2 and they are categorized into four groups as follows.

- Unimodal functions $C1 - C3$, which contain only one global optimal point in the search space.
- Multi-modal functions $C4 - C16$, which have one global optimal point going along with local minimum points.
- Hybrid functions $C17 - C22$: the function variables are randomly divided into some sub-components. Different basic unimodal and multi-modal functions thus are used for the sub-components.
- Composition functions $C23 - C30$ merge the properties of the sub-functions and maintains continuity around the global/local optima.

We carefully carried out experiments with dimension $D \in \{20, 50, 100\}$ for each benchmark function.

B. EVALUATION METHOD

An evaluation method widely used for many algorithms in evolutionary computing is based on the number of iterations, which the algorithms run through. It is notable that in each iteration, GSO-based algorithms run the optimization process in the first phase with m sub-swarms before the second phase, so there are two complete optimization processes taking place in one iteration. This mechanism is different from other swarm-based algorithms since they have only one optimization process for all search agents in each iteration. Therefore, the traditional evaluation method above is no longer in use because it is not fair to estimate the performance of GSO-based algorithms.

In our experiments, we assessed the performance of algorithms with benchmark functions by using several *function evaluations (FE)* [30]. The function evaluation number of an algorithm is merely the number of updating operators over time that the algorithm runs through as follow:

- For PSO and WOA, the number of function evaluations FE is computed by:

$$FE = n * Iter_{max} \tag{18}$$

  where n is the number of particles in population, and max iteration is $Iter_{max}$.
- For GSO-based algorithms:

$$FE = (m.n.l_1 + m.l_2) * Iter_{max} \tag{19}$$

  where m is the sub-swarm number. Each sub-swarm contains n particles; The maximal iteration is $Iter_{max}$. l_1 and l_2 are the iteration of the first and second phase; $m.n.l_1$ and $m.l_2$ are FEs of the first and second phase in each iteration respectively.

With the evaluation method described above, the experimental results of each model are produced by calculating the *mean* value (Equation 20) and the standard deviation value (std, Equation 21) of the resulting series r containing 20 times running for each algorithm.

$$mean = \bar{r} = \frac{1}{n}\sum_{i=1}^{n} r_i \tag{20}$$

$$std = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(r_i - \bar{r})^2} \tag{21}$$

where N is the number of values and r_i $(i = 1, 2, \ldots, N)$ are observations.

TABLE 6. Results and comparison of different algorithms on unimodal and multi-modal benchmark functions with 50D.

Function		GSO	GSO-LWOA	GSO-WOA	LWOA	HGEW
C1	mean	475	100	105	101	**100**
	std	4.68E+02	4.87E-01	5.74E+00	9.31E-01	**4.67E-01**
	rank	5	2	4	3	1
C2	mean	1360	204	254	210	**203**
	std	5.19E+02	4.67E+00	5.63E+01	1.79E+01	**4.36E+00**
	rank	5	2	4	3	1
C3	mean	**300**	301	303	301	300
	std	**1.67E-03**	5.74E-01	2.59E+00	1.22E+00	1.46E-01
	rank	1	3	5	4	2
C4	mean	401	400	400	400	**400**
	std	2.31E+00	3.17E-06	1.66E-06	1.40E-06	**6.12E-07**
	rank	5	4	3	2	1
C5	mean	517	500	500	500	**500**
	std	5.22E+00	6.98E-04	2.14E-03	8.65E-04	**6.12E-04**
	rank	5	2	4	3	1
C6	mean	600	**600**	600	600	600
	std	1.83E-06	**1.11E-08**	1.45E-07	1.74E-08	1.83E-08
	rank	5	1	4	2	3
C7	mean	700	700	700	700	**700**
	std	8.58E-03	4.51E-05	1.38E-04	1.48E-05	**1.06E-05**
	rank	5	3	4	3	1
C8	mean	830	800	800	800	**800**
	std	5.21E+00	3.88E-06	2.78E-05	5.77E-06	**1.26E-06**
	rank	5	2	4	3	1
C9	mean	930	900	900	900	**900**
	std	6.68E+00	9.19E-06	4.37E-05	1.18E-05	**3.00E-06**
	rank	5	2	4	3	1
C10	mean	2540	1000	**1000**	1000	1000
	std	7.26E+02	6.51E-02	**1.26E-03**	5.35E-03	6.55E-02
	rank	5	3	1	2	4
C11	mean	2980	1100	1100	**1100**	1100
	std	1.11E+03	1.58E-02	1.12E-01	**8.23E-05**	1.35E-02
	rank	5	3	4	1	2
C12	mean	1200	1200	**1200**	1200	1200
	std	1.37E-01	9.02E-05	**6.43E-05**	2.72E-04	1.24E-04
	rank	5	2	1	4	3
C13	mean	**1300**	1300	1300	1300	1300
	std	**8.33E-02**	1.08E-01	1.39E-01	2.00E-01	1.49E-01
	rank	1	2	3	5	4
C14	mean	1400	1400	**1400**	1400	1400
	std	1.76E-01	2.02E-01	**1.55E-01**	2.04E-01	2.01E-01
	rank	2	4	1	5	3
C15	mean	1520	1500	1500	**1500**	1500
	std	6.51E+00	6.36E-12	2.64E-11	**3.41E-13**	1.35E-12
	rank	5	3	4	1	2
C16	mean	1600	1600	1600	1600	**1600**
	std	6.24E-01	5.81E-06	8.22E-06	5.23E-06	**2.06E-06**
	rank	5	3	4	2	1
	average rank	4.31	2.56	3.38	2.88	**1.87**
	overall rank	5	2	4	3	**1**

For each function, after *mean* and *std* values of each algorithm are calculated, the algorithm performance is evaluated by ranking in order of the *mean* and *std* values. The evaluation principle is as follows.

- In the case of two algorithms that are as good as each other, their ranks are considered as the average rank of 2 positions, which they account for.
- The *average_rank* for each algorithm is calculated by averaging its positions in all functions.
- The *overall_rank* is determined based on the *average_rank*.

Finally, the rank of the algorithm which shows the best performance in each function is highlighted in bold numbers in Tables 5, 6, 7, 8, 9, and 10.

TABLE 7. Results and comparison of different algorithms on unimodal and multi-modal benchmark functions with 100D.

Function		GSO	GSO-LWOA	GSO-WOA	LWOA	HGEW
C1	mean	32200	101	137	100	**100**
	std	1.70E+04	1.15E+00	3.03E+01	7.54E-03	**2.46E-04**
	rank	5	3	4	2	1
C2	mean	103000	218	335	226	**203**
	std	2.20E+04	1.76E+01	8.24E+01	1.74E+01	**2.34E+00**
	rank	5	2	4	3	1
C3	mean	300	301	**300**	300	300
	std	1.50E-02	7.54E-01	**1.00E-02**	8.33E-02	2.66E-02
	rank	2	5	1	4	3
C4	mean	400	**400**	400	400	400
	std	7.93E-02	**2.67E-09**	7.82E-06	5.19E-08	1.09E-08
	rank	5	1	4	3	2
C5	mean	516	500	**500**	500	500
	std	4.07E+00	1.10E-03	**3.24E-04**	5.91E-04	1.15E-03
	rank	5	3	1	2	4
C6	mean	600	600	600	600	**600**
	std	4.99E-05	9.02E-08	1.16E-07	1.45E-09	**3.79E-10**
	rank	5	3	4	2	1
C7	mean	700	**700**	700	700	700
	std	5.55E-03	**1.54E-06**	3.91E-05	7.99E-06	3.51E-06
	rank	5	1	4	3	2
C8	mean	864	800	800	800	**800**
	std	1.39E+01	1.33E-05	2.46E-05	2.14E-05	**4.10E-06**
	rank	5	4	3	2	1
C9	mean	962	**900**	900	900	900
	std	9.90E-01	**2.41E-06**	6.32E-05	5.88E-06	4.38E-06
	rank	5	1	4	3	2
C10	mean	4800	1000	1000	1000	**1000**
	std	1.19E+03	4.36E-05	2.40E-04	5.39E-05	**1.36E-06**
	rank	5	2	4	3	1
C11	mean	4430	1100	1100	1100	**1100**
	std	1.15E+03	2.32E-04	4.87E-03	4.16E-04	**2.50E-04**
	rank	5	2	4	3	1
C12	mean	1200	1200	1200	1200	**1200**
	std	5.37E-02	1.04E-06	2.07E-05	2.52E-04	**1.54E-08**
	rank	5	2	3	4	1
C13	mean	1300	1300	1300	**1300**	1300
	std	3.18E-02	1.21E-01	1.35E-01	**5.81E-03**	2.37E-02
	rank	3	4	5	1	2
C14	mean	1400	**1400**	1400	1400	1400
	std	1.46E-01	**4.68E-04**	2.22E-01	2.30E-01	1.81E-01
	rank	2	1	4	5	3
C15	mean	1570	1500	1500	1500	**1500**
	std	5.00E+00	6.82E-13	9.09E-13	1.61E-13	**1.61E-13**
	rank	5	2	3	4	1
C16	mean	1610	1600	1600	1600	**1600**
	std	2.77E+00	2.82E-05	9.57E-07	2.91E-06	**7.06E-07**
	rank	5	4	2	3	1
	average rank	4.53	2.47	3.41	2.94	**1.65**
	overall rank	5	2	4	3	**1**

C. SETTING PARAMETERS

The parameters for each algorithms in our experiments after tuning process are set as follows:

- For HGEW, the acceleration constant (c_i) of PSO in the first phase is set to be 2.05 for every c parameter, and with EWOA in the second phase, $p_2 = 0.4$ for balancing exploitation and exploration in EWOA.
- The number of function evaluations is 300000 for all the experiments on benchmark functions with dimensions of 20, 50, 100.

Corresponding parameters set up for GSO-based algorithms and LWOA with the number of functions are shown in Table 3 and Table 4. Note that in Table 3, there are

TABLE 8. Results and comparison of different algorithms on hybrid and composition benchmark functions with 20D.

Function		GSO	GSO-LWOA	GSO-WOA	LWOA	HGEW
C17	mean	1720	1700	1700	**1700**	1700
	std	8.89E+00	9.03E-01	7.72E+00	**1.07E-01**	1.07E+00
	rank	5	2	4	1	3
C18	mean	1870	1810	1820	**1800**	1810
	std	2.94E+01	1.07E+01	3.04E+01	**7.66E+00**	1.29E+01
	rank	5	2	4	1	3
C19	mean	1900	1900	1900	1900	**1900**
	std	1.64E+00	9.42E-01	1.07E+00	1.03E+00	**8.81E-01**
	rank	5	2	4	3	1
C20	mean	2030	2010	2020	2010	**2000**
	std	2.25E+01	6.64E+00	1.91E+01	3.54E+00	**5.00E+00**
	rank	5	3	4	2	1
C21	mean	2110	2100	2100	**2100**	2100
	std	8.69E+00	5.25E-01	6.40E-01	**3.93E-02**	5.98E-01
	rank	5	2	4	1	3
C22	mean	2210	2200	2200	2200	**2200**
	std	4.50E+00	9.44E-01	1.53E+00	4.62E-01	**4.48E-01**
	rank	5	3	4	2	1
C23	mean	**2300**	2300	2300	2300	2300
	std	**3.87E-03**	9.21E-02	1.55E-01	8.26E-02	3.94E-02
	rank	1	4	5	3	2
C24	mean	2570	2400	2400	2400	**2400**
	std	6.46E+01	2.44E-02	8.39E-02	5.35E-02	**1.63E-02**
	rank	5	2	4	3	1
C25	mean	2690	2500	2500	2500	**2500**
	std	5.08E+00	4.37E-02	1.33E-01	5.06E-02	**2.71E-02**
	rank	5	2	4	3	1
C26	mean	2710	2600	2600	2600	**2600**
	std	9.99E+01	6.28E-02	1.82E-01	7.17E-02	**4.28E-02**
	rank	5	2	4	3	1
C27	mean	3150	2710	2710	**2710**	2710
	std	5.29E+01	1.88E-01	2.91E-01	**9.75E-02**	2.09E-01
	rank	5	2	3	1	4
C28	mean	4180	2800	2800	2800	**2800**
	std	3.10E+02	9.30E-02	3.25E-01	8.31E-02	**5.67E-02**
	rank	5	3	4	2	1
C29	mean	3010	2990	2990	**2990**	2990
	std	2.19E+01	1.79E+00	1.08E+01	**6.02E-01**	9.81E-01
	rank	5	3	4	1	2
C30	mean	5190	3000	3010	3000	**3000**
	std	3.65E+03	8.05E-01	1.13E+01	9.37E-01	**5.23E-01**
	rank	5	2	4	3	1
	average rank	4.71	2.43	3.79	**2**	2.07
	overall rank	5	3	4	**1**	2

many sub-iterations of each HGEW and other GSO-based algorithms, as presented in Algorithm 1.

V. EXPERIMENT RESULTS AND DISCUSSION

A. UNIMODAL AND MULTI-MODAL BENCHMARK FUNCTIONS

The achieved results of the tested algorithms are shown in Table 5, 6, and 7. All the experimental results of algorithms on unimodal and multi-modal benchmark functions indicate that HGEW gained the best performance with almost benchmark functions as compared with other optimizers (with descriptions in Section IV). The obtained results stand in the first position, followed by overall and average rankings that are presented in Figure 3. In another perspective, HGEW reaches the optimal global position in many functions regardless of search space size.

1) THE ACCURACY AND THE STABILITY

From the gained results of unimodal functions $C1 - C16$, it could be made the following observations:

- HGEW reaches the optimal values on almost benchmark functions from $C1 - C16$ with decent stability. It is notable that when the dimension of search space increases from 20 to 100, HGEW's performance is still ranked in the first position among all algorithms, while the results of GSO become worse, particularly in $C1$, $C2$, $C10$ and $C11$ (Table 7). Looking more closely at those functions in Table 7, HGEW's mean of best

TABLE 9. Results and comparison of different algorithms on hybrid and composition benchmark functions with 50D.

Function		GSO	GSO-LWOA	GSO-WOA	LWOA	HGEW
C17	mean	2180	1700	1710	1700	**1700**
	std	1.28E+02	1.01E+00	1.14E+01	3.85E-01	**2.50E-01**
	rank	5	3	4	2	1
C18	mean	2680	1810	1860	**1800**	1810
	std	2.30E+02	1.26E+01	6.83E+01	**4.37E+00**	2.84E+01
	rank	5	2	4	1	3
C19	mean	1920	1910	1910	1910	**1900**
	std	1.51E+01	3.20E+00	3.13E+00	1.67E-01	**2.89E+00**
	rank	5	4	3	2	1
C20	mean	2350	2010	2050	**2010**	2010
	std	1.39E+02	7.86E+00	2.98E+01	**1.07E+00**	5.45E+00
	rank	5	3	4	1	2
C21	mean	2150	**2110**	2110	2110	2110
	std	5.56E+01	**2.70E-01**	4.76E+00	5.19E-01	5.44E-01
	rank	5	1	4	2	3
C22	mean	2240	2210	2210	2210	**2200**
	std	1.47E+01	5.44E-01	1.28E+00	3.33E-01	**8.26E-01**
	rank	5	3	4	2	1
C23	mean	2310	2300	2300	2300	**2300**
	std	6.78E-01	9.30E-02	1.84E-01	2.56E-03	**2.34E-03**
	rank	5	3	4	2	1
C24	mean	2960	2400	2400	**2400**	2400
	std	2.22E+02	4.83E-02	1.47E-01	**9.18E-03**	4.11E-02
	rank	5	3	4	1	2
C25	mean	2760	2500	2500	2500	**2500**
	std	1.23E+01	7.98E-02	2.54E-01	2.14E-02	**3.50E-03**
	rank	5	3	4	2	1
C26	mean	2900	2600	2600	2600	**2600**
	std	3.89E+01	1.18E-01	3.61E-01	4.75E-02	**2.88E-02**
	rank	5	3	4	2	1
C27	mean	3230	2710	2710	**2710**	2710
	std	4.87E+01	1.11E-01	4.77E-02	**1.93E-03**	3.69E-02
	rank	5	4	3	1	2
C28	mean	7160	2800	2800	2800	**2800**
	std	4.33E+02	2.18E-02	2.41E-01	4.10E-02	**4.24E-03**
	rank	5	2	4	3	1
C29	mean	3160	**2990**	2990	2990	2990
	std	6.58E+01	**9.60E-03**	5.65E+00	1.27E+00	1.93E-01
	rank	5	1	4	3	2
C30	mean	42000	3010	3010	**3010**	3010
	std	3.88E+04	4.85E+00	5.08E+00	**9.32E-03**	2.86E-01
	rank	5	3	4	1	2
	average rank	5	2.71	3.86	1.79	**1.64**
	average rank	5	3	4	2	**1**

fitness values still reach the optimal values (C1, C10, C11) or nearly the optimal value (C2) with the best stability ($10E-4$), while the others cannot land in the optimal value (C1, C2).

- HGEW outperforms almost other algorithms in term of *std* value on over a half number of functions, especially in $C1, C2, C3, C8, C9$, and $C16$ (Table 6, 7). The results also show that HGEW stays more robust than the others while running on each function 20 times.
- Other algorithms such as LWOA and GSO-WOA bring decent results, being ranked in the second or third place on several functions regardless of the size of the search space. However, they cannot reach the optimal value in unimodal functions ($C1, C2, C3$) with dimensions of 50 and 100 like HGEW.

In summary, all the gained results with unimodal and multimodal functions figure out that HGEW has the excellent optimization ability, and can stay unchanged with the rise in dimension. For explanation, the participation of the current best solution and a random particle in the current swarm, Crossover operator enhances not only the exploitation capacity. It leads the entire swarm to the optimal global position, but also the diversity of the population that helps HGEW escape local minimums.

2) THE CONVERGENCE CHARACTERISTICS

The curve results for unimodal ($C2$) and multi-modal functions ($C15$) are presented in Figure 5 with the dimension rising from 20 to 100 for each functions.

As depicted in the Figure, it is evident that HGEW not only quickly converges towards the optimal global position but

TABLE 10. Results and comparison of different algorithms on hybrid and composition benchmark functions with 100D.

Function		GSO	GSO-LWOA	GSO-WOA	LWOA	HGEW
C17	mean	4560	1700	1700	**1700**	1700
	std	4.73E+01	6.00E-01	5.99E-01	**4.07E-01**	3.05E+00
	rank	5	3	2	1	4
C18	mean	5270	1800	1850	**1800**	1810
	std	4.62E+02	1.94E+00	3.80E+01	**2.75E-01**	6.97E+00
	rank	5	2	4	1	3
C19	mean	1950	1920	1930	1930	**1920**
	std	4.07E-01	4.18E+00	1.36E-02	2.70E-03	**7.00E-01**
	rank	5	2	4	3	1
C20	mean	3860	2010	2020	2010	**2010**
	std	1.03E+01	3.48E-01	8.73E+00	9.37E-03	**1.07E-03**
	rank	5	2	4	3	1
C21	mean	2750	2120	2120	**2120**	2120
	std	2.17E+02	1.93E+00	2.68E+00	**1.91E-02**	1.08E-01
	rank	5	3	4	1	2
C22	mean	2280	2210	2210	2210	**2210**
	std	3.36E+00	2.65E-01	2.32E-01	1.76E-01	**2.77E-02**
	rank	5	3	2	4	1
C23	mean	2600	2310	2310	**2310**	2310
	std	1.08E+01	3.08E-02	2.32E-01	**7.65E-03**	9.39E-02
	rank	5	2	4	1	3
C24	mean	4020	2400	2400	2400	**2400**
	std	2.62E+02	5.08E-02	2.99E-02	7.50E-03	**6.09E-03**
	rank	5	4	3	2	1
C25	mean	2880	2500	2500	**2500**	2500
	std	3.21E-01	5.67E-02	3.72E-02	**1.17E-02**	4.68E-02
	rank	5	4	2	1	3
C26	mean	3070	2600	2600	2600	**2600**
	std	5.09E+01	5.01E-03	8.62E-02	2.67E-02	**1.87E-02**
	rank	5	3	4	2	1
C27	mean	3630	2710	2710	2710	**2710**
	std	1.28E+02	3.21E-01	4.68E-01	2.14E-01	**2.83E-02**
	rank	5	3	4	2	1
C28	mean	14700	2810	2810	2810	**2810**
	std	5.22E+02	1.83E-02	4.69E-01	1.69E-01	**5.02E-03**
	rank	5	2	4	3	1
C29	mean	53100	2990	3010	2990	**2990**
	std	2.09E+04	9.18E-01	1.04E+01	2.13E-01	**7.27E-02**
	rank	5	3	4	2	1
C30	mean	4560	3010	3010	3010	**3010**
	std	1.91E+02	1.53E-01	2.71E-01	8.23E-02	**4.97E-03**
	rank	5	3	4	2	1
	average rank	5	2.86	3.5	2	**1.64**
	overall rank	5	3	4	2	**1**

also provides the best accuracy. Besides, the other GSO-based algorithms show mediocre results in terms of accuracy, even though they perform competitive convergence speed. LWOA also brings good outcomes in some cases and has nearly the same convergence speed compared with the convergence speed of HGEW.

Also, from Figure 5, GSO shows the best performance with the dimension of 20, but when the dimension goes up to 50 or 100, it is outperformed by other algorithms such as HGEW and LWOA. LWOA is competitive in both accuracy and convergence speed, especially in the dimension of 50 in both cases, it shows excellent performance even as decent as HGEW at both unimodal and multi-modal benchmark functions. However, generally, HGEW mostly gives the best results among all algorithms in the term of accuracy.

B. HYBRID AND COMPOSITION BENCHMARK FUNCTIONS

1) THE ACCURACY AND THE STABILITY

To resolve the optimization problem in multi-modal functions $C17 - C30$, it can make an observation that HGEW works very well with the global optimal position in most of the cases as shown in Table 8, 9, and 10. Concretely, HGEW performance is much better than original GSO, LWOA and GSO-WOA with functions $C20$, $C27$ and $C28$ functions. Furthermore, in almost benchmark functions from $C17$ to $C30$, HGEW has the same average *mean* values as compared with other algorithms, especially, for $C16, C17, C19, C22, C24, C25, C26$ functions, it can reach the theoretical optimal position, giving the *std* value at 0.

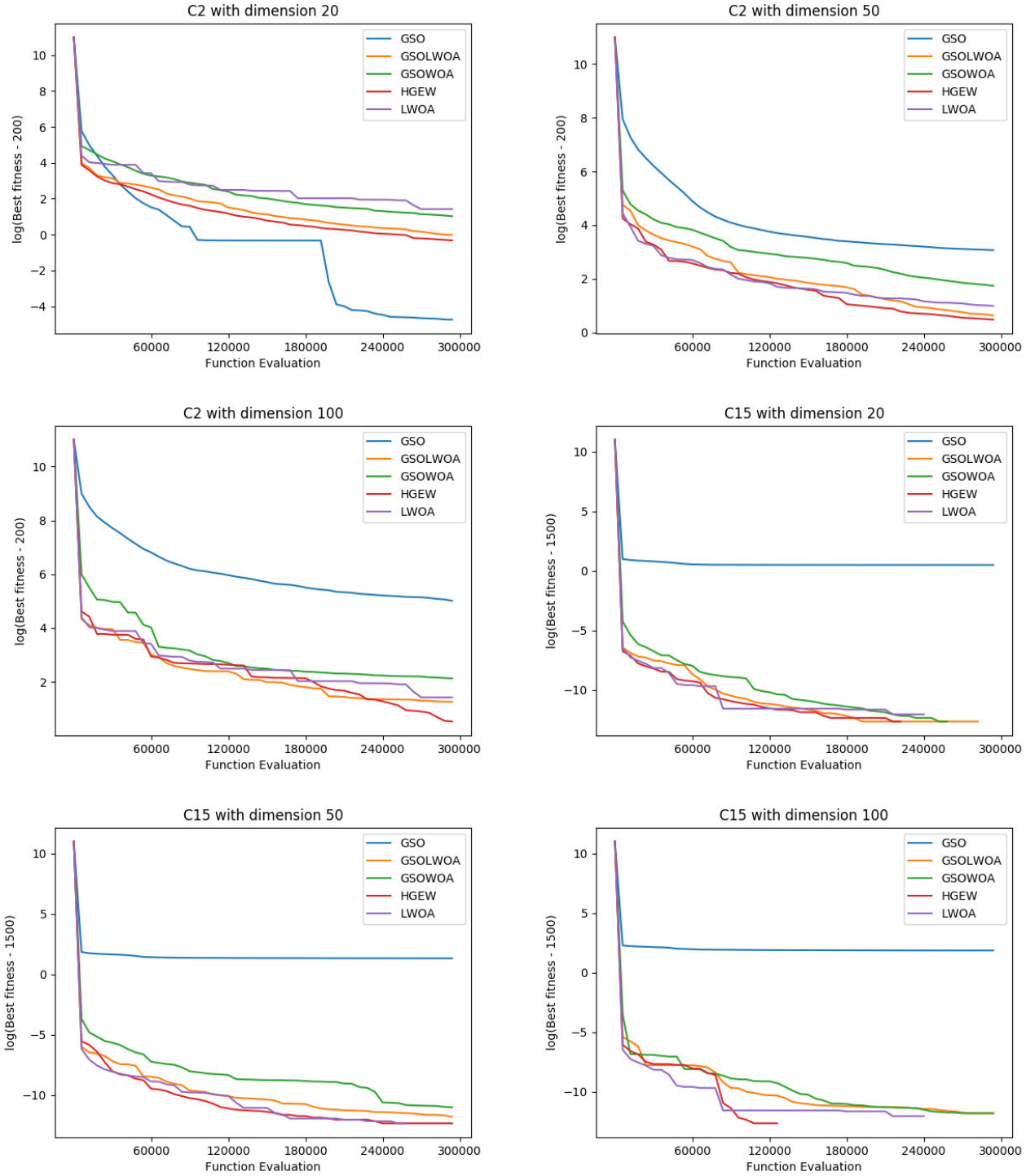


FIGURE 5. The convergence curve of average fitness value on selected unimodal and multi-modal functions.

During the extension of dimension, our HGEW also achieves better results, which are illustrated in Table 10: $C27, C29$ and $C30$ with best fitness value landing very near the optimal values and staying relatively robust (std value $10E-02$). The conclusion is proved through comparisons among the gained outcomes shown in Table 8, and 9. Not only that, but other algorithms such as original GSO or GSO-WOA also generate the worse results as compared with HGEW in terms of *mean* and *std* measurement, specifically, GSO stands in the worst positions in all functions and GSO-WOA is at 4th. As presented in Figure 4, HGEW brings better outcomes following the increment of dimension

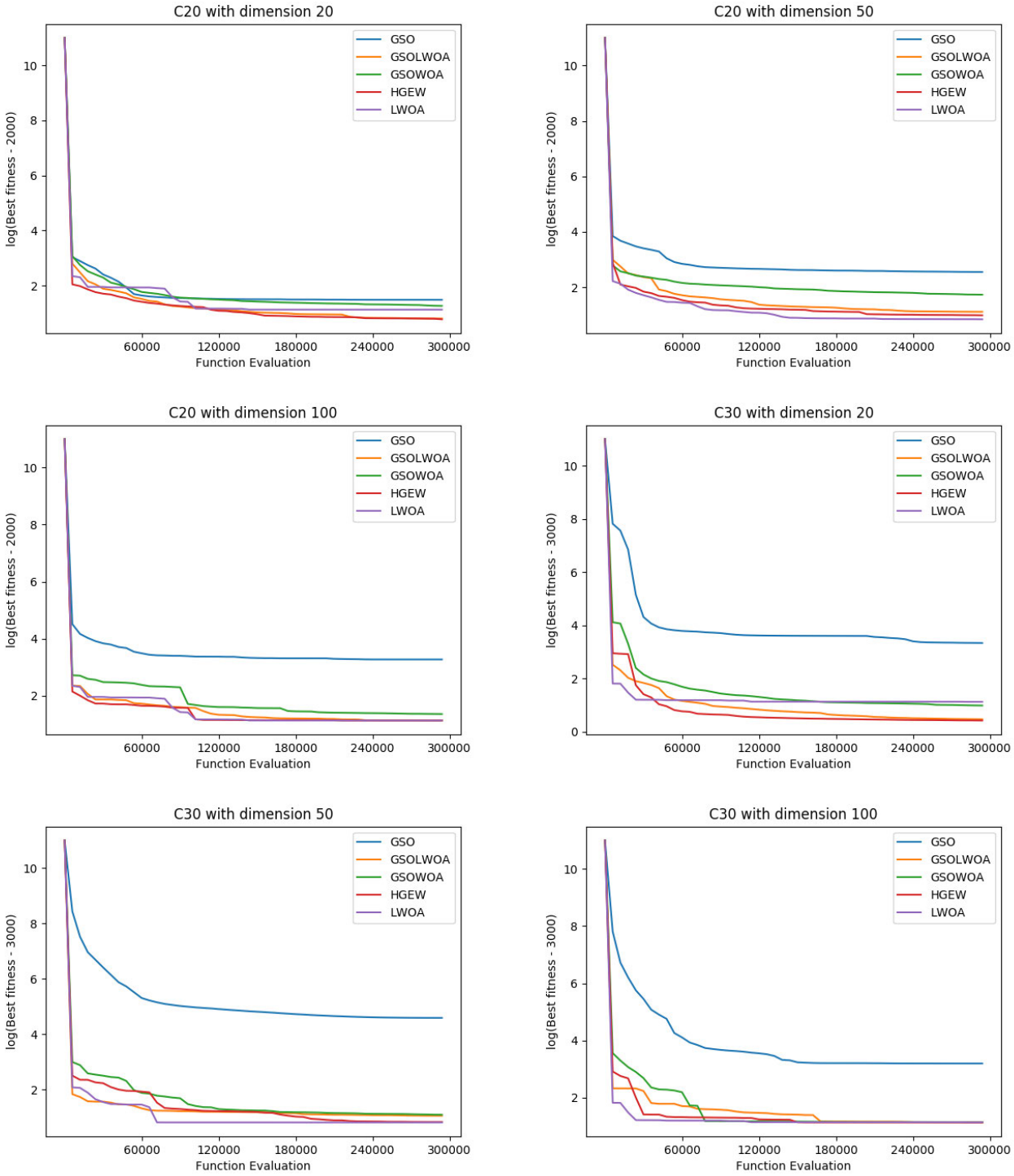


FIGURE 6. The convergence curve of average fitness value on selected hybrid and composition functions.

(average rank is from 2.07 (20D) to 1.64 (50D and 100D), which is the best outcomes among all algorithms), while GSO-LWOA's average ranking becomes worse from 2.43 (20D) to 2.71 and 2.86 (50D and 100D respectively). Thus, HGEW tends to perform a potential solution when the dimension of search space increases; meanwhile, other algorithms become worse. However, LWOA has a promising performance for optimizing function $C29$ while HGEW has the third position in most cases for that function. Even so, HGEW still provides stability in our experiments with outstanding outcomes.

In summary, GSO-LWOA is also very competitive when working on multi-modal functions, but so far behind HGEW in case of unimodal. These gained results proved that our

proposed HGEW could explore the search space efficiently and find promising regions of the search place. In this way, HGEW has a strong ability to accomplish local minimum avoidance as compared with LWOA and GSO-LWOA.

2) THE CONVERGENCE CHARACTERISTICS

The curve results are presented in Figure 6. The dimension is selected in 20D, 50D and 100D with hybrid functions: function $C20$ and composition function $C30$.

As depicted in Figure 6, analogous to the curves in unimodal and multi-modal functions tests, HGEW also generates high convergence speed in hybrid and compos functions with decent accuracy. For and $C20$ functions, HGEW always converges to theoretical global optimal position with acceptable speed compared to LWOA, while for $C20$ function, although HGEW converges after some other algorithms, it performs a better accuracy in the following iterations.

VI. CONCLUSION AND FUTURE WORK

The works presented in this paper aim to tackle the global optimization problem. It is inspired by the simplicity and efficiency principles presented in Galactic Swarm Optimization and Whale Optimization Algorithm. However, with many advantages, the original GSO suffers from premature convergence, and mediocre accuracy, as well as the original WOA, is not optimized in solving global optimization. The HGEW (Hybridization of Galactic Swarm Optimization and Evolution Whale Optimization Algorithm) proposed in this work can overcome those problems with enhancements as follows. Using the Levy-Flight trajectory for adaptive search steps to jump out the optimal local position, and two-point crossover operation for boosting the exploitation capacity in the mean of evolution offspring creation to help the optimizer move more in-depth to the optimal global position in the search space. The main contribution of the work was presented in Section II-E with the evolution hybridization design presented in Section III. The HGEW was validated in Section V with extensive experimental results, comparisons, and evaluations. The obtained results showed that HGEW has an outstanding performance as compared with the others in terms of accuracy and convergence speed.

In terms of the application, HGEW is one of the parts of the V-chain [72] project. In which, the optimizer helps find suitable parameters for neural networks in forecasting workload of distributed systems like cloud computing and blockchain. The main drawback of the proposed technique is similar to other meta-heuristic algorithms. Concretely, although HGEW brings good outcomes, it cannot ensure that the achieved result is the global optimum. Our improvement with Levy-Flight presented in this paper for HGEW also has the goal of enhancing the ability to search the best optima, but we believe that our algorithm's optimization ability can be enhanced even more. Hence, in the future, we try to improve our HGEW by using different methods in the manner of increasing searchability performance as well as to integrate advanced features such as early-stopping condition into our realization.

REFERENCES

[1] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Inf. Sci.*, vol. 295, pp. 407–428, Feb. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025514010251
[2] G. Nguyen, S. Dlugolinsky, M. Bobák, V. Tran, Á. L. García, I. Heredia, P. Malík, and L. Hluchý, "Machine learning and deep learning frameworks and libraries for large-scale data mining: A survey," *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 77–124, Jun. 2019. [Online]. Available: https://link.springer.com/article/10.1007%2Fs10462-018-09679-z
[3] G. Nguyen, S. Dlugolinsky, V. Tran, and A. Lopez Garcia, "Deep learning for proactive network monitoring and security protection," *IEEE Access*, vol. 8, pp. 19696–19716, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8966259
[4] H. N. Bessenouci, Z. Sari, and L. Ghomri, "Metaheuristic based control of a flow rack automated storage retrieval system," *J. Intell. Manuf.*, vol. 23, no. 4, pp. 1157–1166, Aug. 2012. [Online]. Available: https://link.springer.com/article/10.1007/s10845-010-0432-1
[5] D. K. Sambariya and R. Prasad, "Robust tuning of power system stabilizer for small signal stability enhancement using metaheuristic bat algorithm," *Int. J. Electr. Power Energy Syst.*, vol. 61, pp. 229–238, Oct. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0142061514001616
[6] E.-G. Talbi, "Combining metaheuristics with mathematical programming, constraint programming and machine learning," *Annals of Operations Research*, vol. 11, no. 2, pp. 101–150, Jul. 2013. [Online]. Available: https://link.springer.com/article/10.1007/s10479-015-2034-y
[7] S. Ramírez-Gallego, A. Fernández, S. García, M. Chen, and F. Herrera, "Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce," *Inf. Fusion*, vol. 42, pp. 51–61, Jul. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253517305912
[8] A. Cano and B. Krawczyk, "Learning classification rules with differential evolution for high-speed data stream mining on GPU s," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8477961
[9] M. Liu, F. Zhang, Y. Ma, H. R. Pota, and W. Shen, "Evacuation path optimization based on quantum ant colony algorithm," *Adv. Eng. Informat.*, vol. 30, no. 3, pp. 259–267, Aug. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034616300660
[10] H. Chen, Y. Xu, M. Wang, and X. Zhao, "A balanced whale optimization algorithm for constrained engineering design problems," *Appl. Math. Model.*, vol. 71, pp. 45–59, Jul. 2019, doi: 10.1016/j.apm.2019.02.004.
[11] A. A. El-Fergany, H. M. Hasanien, and A. M. Agwa, "Semi-empirical PEM fuel cells model using whale optimization algorithm," *Energy Convers. Manage.*, vol. 201, Dec. 2019, Art. no. 112197, doi: 10.1016/j.enconman.2019.112197.
[12] J. Luo, H. Chen, A. A. Heidari, Y. Xu, Q. Zhang, and C. Li, "Multi-strategy boosted mutative whale-inspired optimization approaches," *Appl. Math. Model.*, vol. 73, pp. 109–123, Sep. 2019, doi: 10.1016/j.apm.2019.03.046.
[13] M. A. Elaziz and S. Mirjalili, "A hyper-heuristic for improving the initial population of whale optimization algorithm," *Knowl.-Based Syst.*, vol. 172, pp. 42–63, May 2019, doi: 10.1016/j.knosys.2019.02.010.
[14] A. A. Heidari, I. Aljarah, H. Faris, H. Chen, J. Luo, and S. Mirjalili, "An enhanced associative learning-based exploratory whale optimizer for global optimization," *Neural Comput. Appl.*, vol. 32, no. 9, pp. 5185–5211, May 2020, doi: 10.1007/s00521-019-04015-0.
[15] E.-G. Talbi, *Metaheuristics: From Design to Implementation*, vol. 74. Hoboken, NJ, USA: Wiley 2009. [Online]. Available: https://www.wiley.com/en-us/Metaheuristics%3A+From+Design+to+Implementation+-p-9780470278581
[16] E. Alba and B. Dorronsoro, "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 126–142, Apr. 2005. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1413255
[17] M. Lozano and C. García-Martínez, "Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report," *Comput. Oper. Res.*, vol. 37, no. 3, pp. 481–497, Mar. 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054809000471

[18] T. Nguyen, T. Nguyen, B. M. Nguyen, and G. Nguyen, "Efficient time-series forecasting using neural network and opposition-based coral reefs optimization," *Int. J. Comput. Intell. Syst.*, vol. 12, no. 2, p. 1144, 2019. [Online]. Available: https://www.atlantis-press.com/journals/ijcis/125921354

[19] G. Nguyen, B. M. Nguyen, D. Tran, and L. Hluchy, "A heuristics approach to mine behavioural data logs in mobile malware detection system," *Data Knowl. Eng.*, vol. 115, pp. 129–151, May 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169023X17303063?via%%3Dihub

[20] S. Mirjalili and J. S. Dong, "Introduction to nature-inspired algorithms," in *Nature-Inspired Optimizers*. Cham, Switzerland: Springer, 2020, pp. 1–5. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-12127-3_1

[21] S. Ólafsson, "Metaheuristics," *Handbooks Oper. Res. Manage. Sci.*, vol. 13, pp. 633–654, Aug. 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927050706130212

[22] 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. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8599578

[23] 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.* Kitakyushu, Japan: Springer, 2019, pp. 501–517. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-14812-6_31

[24] J. Zhang, M. Xiao, L. Gao, and Q. Pan, "Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems," *Appl. Math. Model.*, vol. 63, pp. 464–490, Nov. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0307904X18302890

[25] A. A. Heidari and P. Pahlavani, "An efficient modified grey wolf optimizer with Lévy flight for optimization tasks," *Appl. Soft Comput.*, vol. 60, pp. 115–134, Nov. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494617303873

[26] M. Mafarja, I. Aljarah, A. A. Heidari, A. I. Hammouri, H. Faris, A. M. Al-Zoubi, and S. Mirjalili, "Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems," *Knowl.-Based Syst.*, vol. 145, pp. 25–45, Apr. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705117306159

[27] M. Mafarja, I. Aljarah, A. A. Heidari, H. Faris, P. Fournier-Viger, X. Li, and S. Mirjalili, "Binary dragonfly optimization for feature selection using time-varying transfer functions," *Knowl.-Based Syst.*, vol. 161, pp. 185–204, Dec. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095070511830399X

[28] I. Aljarah, M. Mafarja, A. A. Heidari, H. Faris, Y. Zhang, and S. Mirjalili, "Asynchronous accelerating multi-leader Salp chains for feature selection," *Appl. Soft Comput.*, vol. 71, pp. 964–979, Oct. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494618304289

[29] J. Kennedy, "Particle swarm optimization," *Encyclopedia Mach. Learn.*, vol. 4, pp. 760–766, Nov. 2010. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/488968

[30] V. Muthiah-Nakarajan and M. M. Noel, "Galactic swarm optimization: A new global optimization metaheuristic inspired by galactic motion," *Appl. Soft Comput.*, vol. 38, pp. 771–787, Jan. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494615006742

[31] N. Pholdee and S. Bureerat, "A comparative study of eighteen self-adaptive metaheuristic algorithms for truss sizing optimisation," *KSCE J. Civil Eng.*, vol. 22, no. 8, pp. 2982–2993, Aug. 2018. [Online]. Available: https://link.springer.com/article/10.1007/s12205-017-0095-y

[32] E. Bernal, O. Castillo, J. Soria, and F. Valdez, "Optimization of fuzzy controller using galactic swarm optimization with Type-2 fuzzy dynamic parameter adjustment," *Axioms*, vol. 8, no. 1, p. 26, 2019. [Online]. Available: https://www.mdpi.com/2075-1680/8/1/26

[33] S. J. Nanda, I. Gulati, R. Chauhan, R. Modi, and U. Dhaked, "A K-means-galactic swarm optimization-based clustering algorithm with Otsu's entropy for brain tumor detection," *Appl. Artif. Intell.*, vol. 33, no. 2, pp. 152–170, Jan. 2019. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1/08839514.2018.1530869

[34] C. Bagchi, D. G. B. Amali, and M. Dinakaran, "Accurate facial ethnicity classification using artificial neural networks trained with galactic swarm optimization algorithm," in *Information Systems Design and Intelligent Applications*. Singapore: Springer, 2019, pp. 123–132. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-13-3329-3_12

[35] S. N. Qasem and M. Alsaidan, "A new hybrid intelligent system for prediction of medical diseases," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 4, pp. 366–379, 2018. [Online]. Available: https://pdfs.semanticscholar.org/4ac6/93918f94996fc1c377ca7dc1a17638a2b%719.pdf

[36] E. Bernal, O. Castillo, J. Soria, and F. Valdez, "Galactic swarm optimization with adaptation of parameters using fuzzy logic for the optimization of mathematical functions," in *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications*. Cham, Switzerland: Springer, 2018, pp. 131–140. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-71008-2_11

[37] E. Bernal, O. Castillo, J. Soria, and F. Valdez, "Fuzzy galactic swarm optimization with dynamic adjustment of parameters based on fuzzy logic," *Metaheuristics*, pp. 1–19, Dec. 2018. [Online]. Available: https://link.springer.com/article/10.1007/s42979-020-0062-4

[38] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons Fractals*, vol. 25, no. 5, pp. 1261–1271, Sep. 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960077905000330

[39] E. Kaya, S. A. Uymaz, and B. Kocer, "Boosting galactic swarm optimization with ABC," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 9, pp. 2401–2419, Sep. 2019. [Online]. Available: https://link.springer.com/article/10.1007/s13042-018-0878-6

[40] W. Guo, T. Liu, F. Dai, and P. Xu, "An improved whale optimization algorithm for forecasting water resources demand," *Appl. Soft Comput.*, vol. 86, Jan. 2020, Art. no. 105925, doi: 10.1016/j.asoc.2019.105925.

[41] M. Abdel-Basset, G. Manogaran, D. El-Shahat, and S. Mirjalili, "A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem," *Future Gener. Comput. Syst.*, vol. 85, pp. 129–145, Aug. 2018, doi: 10.1016/j.future.2018.03.020.

[42] L. Jain, R. Katarya, and S. Sachdeva, "Opinion leader detection using whale optimization algorithm in online social network," *Expert Syst. Appl.*, vol. 142, Mar. 2020, Art. no. 113016, doi: 10.1016/j.eswa.2019.113016.

[43] M. Wang and H. Chen, "Chaotic multi-swarm whale optimizer boosted support vector machine for medical diagnosis," *Appl. Soft Comput.*, vol. 88, Mar. 2020, Art. no. 105946, doi: 10.1016/j.asoc.2019.105946.

[44] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965997816300163

[45] W. A. Watkins and W. E. Schevill, "Aerial observation of feeding behavior in four baleen whales: Eubalaena glacialis, balaenoptera borealis, megaptera novaeangliae, and balaenoptera physalus," *J. Mammal.*, vol. 60, no. 1, pp. 155–163, Feb. 1979. [Online]. Available: https://academic.oup.com/jmammal/article/60/1/155/896329

[46] Y. Sun, X. Wang, Y. Chen, and Z. Liu, "A modified whale optimization algorithm for large-scale global optimization problems," *Expert Syst. Appl.*, vol. 114, pp. 563–577, Dec. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417418305360

[47] D. Tran, N. Tran, G. Nguyen, and B. M. Nguyen, "A proactive cloud scaling model based on fuzzy time series and SLA awareness," *Procedia Comput. Sci.*, vol. 108, pp. 365–374, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050917306865

[48] L. Hluchy, G. Nguyen, J. Astaloš, V. Tran, V. Šipková, and B. M. Nguyen, "Effective computation resilience in high performance and distributed environments," *Comput. Informat.*, vol. 35, no. 6, pp. 1386–1415, 2017. [Online]. Available: http://www.cai.sk/ojs/index.php/cai/article/view/3537/802

[49] B. M. Nguyen, H. T. T. Binh, T. T. Anh, and D. B. Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment," *Appl. Sci.*, vol. 9, no. 9, p. 1730, Apr. 2019, doi: 10.3390/app9091730.

[50] Y. Jiang, T. Hu, C. Huang, and X. Wu, "An improved particle swarm optimization algorithm," *Appl. Math. Comput.*, vol. 193, no. 1, pp. 231–239, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S009630030700392X

[51] G. Kaur and S. Arora, "Chaotic whale optimization algorithm," *J. Comput. Design Eng.*, vol. 5, no. 3, pp. 275–284, Jul. 2018.

[52] Y. Ling, Y. Zhou, and Q. Luo, "Lévy flight trajectory-based whale optimization algorithm for global optimization," *IEEE Access*, vol. 5, pp. 6168–6186, 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7904636

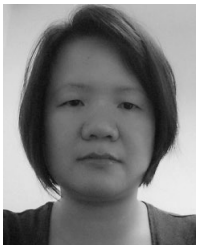
[53] I. N. Trivedi, J. Pradeep, J. Narottam, K. Arvind, and L. Dilip, "Novel adaptive whale optimization algorithm for global optimization," *Indian J. Sci. Technol.*, vol. 9, no. 38, pp. 319–326, 2016. [Online]. Available: https://tarjomefa.com/wp-content/uploads/2017/12/TarjomeFa-F379-English%.pdf

[54] A. Kaveh and M. I. Ghazaan, "Enhanced whale optimization algorithm for sizing optimization of skeletal structures," *Mech. Based Design Struct. Mach.*, vol. 45, no. 3, pp. 345–362, Jul. 2017. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/15397734.2016.1213639
[55] G. Xiong, J. Zhang, D. Shi, and Y. He, "Parameter extraction of solar photovoltaic models using an improved whale optimization algorithm," *Energy Convers. Manage.*, vol. 174, pp. 388–405, Oct. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890418309026
[56] D. Oliva, M. A. El Aziz, and A. E. Hassanien, "Parameter estimation of photovoltaic cells using an improved chaotic whale optimization algorithm," *Appl. Energy*, vol. 200, pp. 141–154, Aug. 2017.
[57] X.-S. Yang, "Firefly algorithm, Lévy flights and global optimization," in *Research and Development in Intelligent Systems XXVI*. London, U.K.: Springer, 2010, pp. 209–218. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-84882-983-1_15
[58] G. Wang, L. Guo, A. H. Gandomi, L. Cao, A. H. Alavi, H. Duan, and J. Li, "Lévy-flight krill herd algorithm," *Math. Problems Eng.*, vol. 2013, Feb. 2013, Art. no. 682073. [Online]. Available: https://www.hindawi.com/journals/mpe/2013/682073/
[59] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965997813001853
[60] A. Korashy, S. Kamel, F. Jurado, and A.-R. Youssef, "Hybrid whale optimization algorithm and grey wolf optimizer algorithm for optimal coordination of direction overcurrent relays," *Electr. Power Compon. Syst.*, vol. 47, nos. 6–7, pp. 644–658, Apr. 2019, doi: 10.1080/15325008.2019.1602687.
[61] S. Arora and S. Singh, "Butterfly algorithm with Lévy flights for global optimization," in *Proc. Int. Conf. Signal Process., Comput. Control (ISPCC)*, Sep. 2015, pp. 220–224. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7375029
[62] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18313530
[63] J. H. Holland, "Genetic algorithms," *Sci. Amer.*, vol. 267, no. 1, pp. 66–73, 1992. [Online]. Available: https://www.cc.gatech.edu/~turk/bio_sim/articles/genetic_algorithm.pdf
[64] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/294849
[65] 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, pp. 341–359, 1997. [Online]. Available: https://link.springer.com/article/10.1023/A:1008202821328
[66] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput. IEEE World Congr. Comput. Intell.*, May 1998, pp. 69–73. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/699146
[67] T. Tran. (2020). *HGEW Implementation Code*. Accessed: Apr. 5, 2020. [Online]. Available: http://dx.doi.org/10.5281/zenodo.3703161
[68] H. M. Mohammed, S. U. Umar, and T. A. Rashid, "A systematic and meta-analysis survey of whale optimization algorithm," *Comput. Intell. Neurosci.*, vol. 2019, pp. 1–25, Apr. 2019. [Online]. Available: https://www.hindawi.com/journals/cin/2019/8718571/
[69] T. Nguyen, "A collection of benchmark functions for numerical optimization problems. Framework of optimization function in NUmpy (opfunu)," Hanoi Univ. Sci. Technol., Hanoi, Vietnam, Tech. Rep. Version 1.0, Mar. 2020, doi: 10.5281/zenodo.3620960.
[70] J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou China, Nanyang Technol. Univ., Singapore, Tech. Rep. 201311, 2013, vol. 635. [Online]. Available: https://bee22.com/resources/Liang%20CEC2014.pdf
[71] CEC. (2020). *Benchmarks for Evaluation of Evolutionary Algorithms*. Accessed: Apr. 5, 2020. [Online]. Available: https://www.ntu.edu.sg/home/epnsugan/index_files/cec-benchmarking.htm
[72] B. Group. (2020). *V-Chain: A Blockchain-Based Platform for Development and Deployment of Decentralized Applications*. Accessed: Apr. 5, 2020. [Online]. Available: https://v-chain.vn/

BINH MINH NGUYEN received the Dipl.Ing. degree in computer-aided design from the Institute of Automation and Information Technologies, Tambov State Technical University (TSTU), Russia, in 2008, and the Ph.D. degree in applied informatics from the Faculty of Informatics and Information Technology, Slovak University of Technology (STU), Slovakia, in 2013. From 2008 to 2013, he worked as a Researcher with the Institute of Informatics, Slovak Academy of Sciences (IISAS), Slovakia. He is currently a Lecturer and the Head of the Department of Information Systems, School of Information and Communication Technology (SoICT), Hanoi University of Science and Technology (HUST), Vietnam. His research interests include distributed systems and data analytics for several application domains, including cloud services, fog computing, and blockchain.

TRUNG TRAN studies computer science program with the School of Information and Communication Technology (SoICT), Hanoi University of Science and Technology, Vietnam. In 2014, he won the Second Prize of the Annual Mathematical Competition for high school students in Hanoi. He also was awarded the position once again, in 2015. He also participated in the Scientific Research Contest for students, in 2019, and his work belongs to the best ten research works in the university. He is also interested in evolutionary computation, optimization, and data analytics.

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). In 2013, he was awarded the Third Prize in Mathematical Competition for High School students at the provincial level. In recent two years, he contributes as the first author to several articles in international conferences and journals with research topics focused on intelligent computational methods using neural networks and meta-heuristic algorithms for proactive forecasting in cloud resource management.



GIANG NGUYEN received the M.Sc. degree in informatics and information technology and the Ph.D. degree in applied informatics from the Slovak University of Technology (STU), Bratislava, Slovakia. She is currently a Senior Researcher with the Institute of Informatics, Slovak Academy of Sciences (IISAS), with research topics focused on soft computing, machine learning, deep learning, and high-performance computing. She is also a coauthor of scientific articles and has been participating in EU IST RTD, EU H2020 projects, and national projects. She is also a Supervisor for a Ph.D. study at IISAS, FIIT STU, FMFI UK, Bratislava, and FEI TUKE, Košice, Slovakia.