Artificial Intelligence Review (2023) 56:9837–9947
https://doi.org/10.1007/s10462-022-10309-y

A new hyper-heuristic based on ant lion optimizer and Tabu search algorithm for replica management in cloud environment

Behnam Mohammad Hasani Zade[1] · Najme Mansouri[1] · Mohammad Masoud Javidi[1]

Published online: 23 November 2022

Abstract
Information can be shared across the Internet using cloud computing, a powerful paradigm for meeting the needs of individuals and organizations. To minimize access time and maximize load balancing for data nodes (DNs), a dynamic data replication algorithm is necessary. Even so, few of the existing algorithms consider each objective holistically during replication. An improved ant lion optimizer (ALO) algorithm and a fuzzy system are used in this paper to determine dynamically the number of replicas and the DNs for replication. Further, it balances the trade-offs among different objectives (e.g., service time, system availability, load, and monetary cost). The ALO algorithm has been widely applied to solve complex optimization problems due to its simplicity in implementation. However, ALO has premature convergence and can thus easily get trapped into the local optimum solution. In this paper, to overcome the shortcomings of ALO by balancing exploration and exploitation, a hybrid ant lion optimizer with Tabu search algorithm (ALO-Tabu) is proposed. There are several improvements of the ALO, in which the appropriate solutions are selected for the initial population based on chaotic maps (CMs) and opposition-based learning (OBL) strategies. On the other hand, there are many CMs, OBLs, and random walk strategies that make it difficult to select the best one for optimization. Generally, they are selected manually, which is time-consuming. As a result, this paper presents a hyper-heuristic ALO (HH-ALO-Tabu) that automatically chooses CMs, OBLs, and random walk strategies depending on the differential evolution (DE) algorithm. Based on 20 well-known test functions, the experiment results and statistical tests show that HH-ALO-Tabu can solve optimization problems effectively.

Keywords Cloud computing · Data replication · Ant lion optimizer · Chaotic · Fuzzy system · Meta-heuristics

✉ Najme Mansouri
najme.mansouri@gmail.com

[1] Department of Computer Science, Shahid Bahonar University of Kerman, Box No. 76135-133, Kerman, Iran

1 Introduction

The cloud is a distributed computing system in which applications and computing infrastructure are provided as a service. Data replication techniques are needed for cloud computing with data-intensive workflows. Cloud computing has emerged with broad-ranging effects across business organizations, software engineering, information technology, and data storage (Mansouri et al. 2020; Han et al. 2021; Mohammad Hasani Zade et al. 2021). For example, the main effects are the enhancement of their capability, cost savings, better mobility, and flexibility. Cloud computing considers paying as you go model. In other words, you pay for the services you have used. Gartner et al. (Kwame Senyo et al. 2018) mentioned that more than $1 trillion of IT expenditure is directly or indirectly related to cloud systems in 2020. In addition, they estimate worldwide end-user spending on public cloud services grows 23.1% in 2021. Recently, there is strong competition among the main providers of cloud services like Amazon, Microsoft, and Google for a share of this projected revenue. Table 1 presents the five essential features of cloud computing.

1.1 Data replication

Scientific disciplines are increasingly dependent on data sets as a result of technological advancements. Because researchers and practitioners are located across geographical regions, large data sets are usually stored in data nodes (DNs) of cloud environments. These distributed data must be available and accessible with high performance. Generally, replicas are created in different locations to address these challenges.

There have been many applications of data replication in recent years, including the World Wide Web, peer-to-peer networks, ad hoc networks, sensor networks, mesh networks, and data grids (Alami Milani and Jafari Navimipour 2016). More recently, in the cloud (Yao et al. 2018; Mansouri and Javidi 2018; Mansouri 2016; Wu 2017; Khalili Azimi 2019a) data replication has become one of the prominent topics once again due to complex scientific applications of the cloud need to process mass data. It is possible to increase reliability, reduce waiting times for users, increase data availability, and optimize resource consumption through the use of a cloud data replication strategy.

There are two types of data replication algorithms: static and dynamic. The static replication algorithm preconfigures the number of replicas before starting the system and these copies are not adapted based on the changes in the environment. A dynamic data replication considers the dynamic nature of the cloud and decides what, when, and where to replicate based on the changing trends of the cloud. Due to node additions and removals, and changes in file access patterns over time, dynamic replication algorithms must be

Table 1 Main characteristics of cloud computing (Branco et al. 2017)

Characteristic	Description
On-demand self-service	Provisioning automatically
Broad network access	Easily accessible via any networked device
Resource pooling	A multi-tenancy system with a location-independent model
Rapid elasticity	Scaling of loads automatically
Measured service	Monitoring, reporting, and billing

periodically reconfigured. An individual's file access patterns indicate how they access data for purpose of completing tasks.

When a certain threshold is reached, a dynamic replication algorithm replicates the popular file in the best location. Furthermore, each historical data access record can be assigned a different weight, which differentiates the importance of each record. As a result, the system must record the history of end-to-end data transfers in order to make adaptive decisions during operation. Nevertheless, it is difficult to collect the run time information of all files in complex cloud infrastructure, and hence one of the essential parts of cloud structure is cloud information service (CIS) which stores all necessary information about all DNs in the cloud environment.

As shown in Fig. 1, static and dynamic replication algorithms are compared. Deciding the number of replicas replica placement in large heterogeneous systems is more difficult compared to small environments, where each site may have various abilities. Ineffective replica placement may cause the access skew to some busy sites but some other sites are idle and hence result in low performance and load imbalance across the cloud system. There are two important questions raised by this significant deficiency:

(1) When these replicas are placed, how does this improve the performance of the system?
(2) Is it necessary to create a certain number of replicas to maintain good system availability?

Dynamically determining the number and location of replicas is accomplished using a hyper-heuristic ant lion-Tabu replication algorithm coupled with a fuzzy system. The best solution of the proposed method simultaneously identifies the suitable number of replicas for each candidate file and their best location by taking into account several parameters [i.e., service time, system availability (SA), load, and cost].

A distributed system has NP-hard problems for achieving optimal data replication (Tang et al. 2005; Du et al. 2011). Most of the previous data replication algorithms are not very suitable for the cloud environment because they focus only on performance (i.e., fast response). While one of the key concerns of providers is maximizing their profit as well as satisfying tenant requirements. Several factors play a role in the decision to replicate data, including file availability, service time, energy consumption, and the profit of the provider. Data replication algorithms with some conflict objectives try to excel in at least one Quality of Service (QoS) parameter with minimal undesirable consequences. In this regard, meta-heuristic techniques are useful to find a near-optimal solution for replica configuration

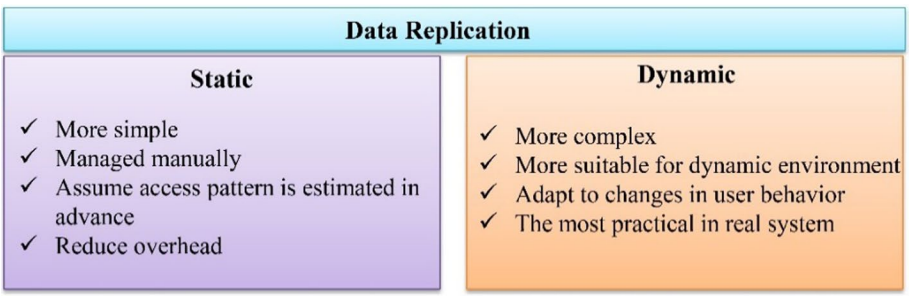

Fig. 1 Static vs. dynamic replication

(Mahdavi Jafari and Khayati 2018; Parejo et al. 2012). Some of the recent meta-heuristic algorithms are grey wolf optimizer, GWO (Mirjalili et al. 2014), magnetic charged system search (Kaveh et al. 2013), ray optimization, RO (Kaveh and Khayatazad 2013) algorithm, colliding bodies optimization, CBO (Kaveh and Mahdavi 2014) algorithm, hybrid particle swallow swarm optimization, HPSSO (Kaveh et al. 2014), democratic particle swarm optimization, DPSO (Kaveh and Zolghadr 2014), dolphin echolocation, DE (Kaveh 2014), grasshopper optimization algorithm, GOA (Saremi et al. 2017), and whale optimization algorithm, WOA (Mirjalili and Lewis 2016).

The arithmetic optimization algorithm (AOA) proposed by Abualigah et al. (Abualigah et al. 2021a) uses the distribution behavior of arithmetic operators (multiplication, division, subtraction, and addition). In (Abualigah et al. 2021b), a novel Aquila optimizer is presented to solve complex problems such as thirty CEC 2017 test functions. Despite this, these algorithms cannot solve any problem of optimization, so the hybrid method offers a greater chance of success. For example, Abualigah and Dulaimi (Abualigah and Dulaimi 2021) introduced a hybrid sine cosine algorithm and genetic algorithm for feature selection problems. A steady-state genetic algorithm (SSGA) with exploration and exploitation capabilities was combined by Mousa et al. (Mousa et al. 2020). A performance and application evaluation is provided in Abualigah and Diabat (2021).

In this paper, we select ant lion optimizer (ALO) algorithm to determine the number of replicas and the best location for them in the cloud (Mirjalili 2015). Several real-world problems can be solved using the ALO algorithm, such as feature selection, scheduling, and structure design. Experiment results in Mirjalili (2015) demonstrated that the ALO algorithm provides very competitive results in terms of exploration, exploitation, local optima avoidance, and convergence speed with dynamic stochastic selection (DEDS) (Zhang et al. 2008), hybrid particle swarm optimization with differential evolution, PSODE (Liu et al. 2010) mine blast algorithm, MBA (Sadollah et al. 2013), and cuckoo search, CS (Gandomi et al. 2013a). High exploitation of the ALO algorithm can be concluded from adaptive boundary shrinking procedure and elitism. The random walk and roulette wheel selection (RWS) methods make it highly exploratory.

Additionally, we use a fuzzy system to design the multi-objective replication algorithm. Fuzzy systems are found to be superior when dealing with complex decisions involving multiple objectives, allowing the decision-maker to make an accurate, easy decision when facing multi-parameter decisions. It seems that fuzzy tools provide more efficient results than existing techniques using approximate reasoning and linguistic terms. Their role becomes more intense when used to complex phenomena not easily explained by traditional mathematics. As a result, a fuzzy rule-based system within the cloud is currently gaining interest due to the cloud's inherent dynamic and complex nature. In the replication process, fuzzy rule-based systems can show system knowledge and the interaction among important variables such as service time, SA, load, and cost (Wang et al. 2017; Beigrezaei et al. 2021).

1.2 Contributions

The majority of replication algorithms previously implemented only focused on achieving performance expectations without affecting the provider's profit. The proposed algorithm is designed to guarantee the provider's economic profit while satisfying the user's needs. In this regard, we have a multi-objective optimization problem that consists of optimizing a set of conflict objectives.

By increasing the number of replicas, for example, low access time and high availability improve. To minimize energy and financial costs, the number of replicas should be as small as possible. Therefore, the aim is to achieve good trade-off solutions that show the best possible compromises among the objectives. To provide Pareto optimal solutions, meta-heuristic techniques are most commonly used to solve optimization problems.

As a result of using the ALO algorithm in this paper, we will be able to significantly improve the optimization quality of the data replication problem. Some improvements are introduced into the ALO algorithm before it is employed for data replication (i.e., using a chaotic function, an OBL strategy, and improved ant lion random walks), and this enhanced strategy is called ALO-Tabu. To find the nearly optimal configuration, we use a hyper-heuristic version of the proposed ALO-Tabu. After an optimal configuration for the chaotic maps (CMs), OBL strategies, and the random walks is selected based on the DE algorithm, this optimal configuration is used to improve the performance of the proposed ALO-Tabu in finding the optimal solution to the given problem. Based on our presented strategy, we were able to contribute the following.

(1) The DE algorithm finds the optimal configuration from several chaotic functions, OBL strategies, and random walks for improving ALO.
(2) By selecting the chaos function and OBL strategy, the quality of the initial population has been improved.
(3) The exploitation of the algorithm has been increased by Tabu search (TS). A hybrid method consisting of Tabu and ALO has been proposed in which TS helps the proposed method to apply different random walks so the ants can easily escape from a local optimal.
(4) The antlions in the ALO algorithm are moved (relocated) in a novel way based on the number of iteration, their fitness, and corresponding ants in the ant population.
(5) Two well-known groups of benchmark functions including 10 test functions from CEC 2005 and 10 test functions from CEC 2019 are used to demonstrate the effectiveness of the proposed method called hyper-heuristic ant lion-Tabu (HH-ALO-Tabu) in finding global optimal.
(6) The proposed HH-ALO-Tabu algorithm is compared with two categories of methods. The first set includes some state-of-the-art optimization algorithms and the second set consists of ALO variants.
(7) A new replication method (FHHAT) based on the proposed HH-ALO-Tabu is introduced. The FHHAT algorithm models four conflicting objectives as SA, load, mean service time, and total costs of the provider simultaneously in determining the number of replicas and placing new replicas.
(8) The proposed FHHAT algorithm uses the fuzzy inference system (FIS) for fitness evaluation in the replication problem.
(9) Using the CloudSim (Goyal et al. 2012) toolkit, experiments indicated that FHHAT had a better performance in terms of replications, load variance, average service time, availability, and total costs of the provider compared with multi-objective optimized replication management, MORM (Long et al. 2014), multi-objective evolutionary, MOE (Hassan et al. 2009), dynamic popularity aware replication strategy, DPRS (Mansouri et al. 2017), and prefetching-aware data replication, PDR (Mansouri and Javidi xxxx) algorithms.

Figures 2 and 3 describe the contributions in schematic form.

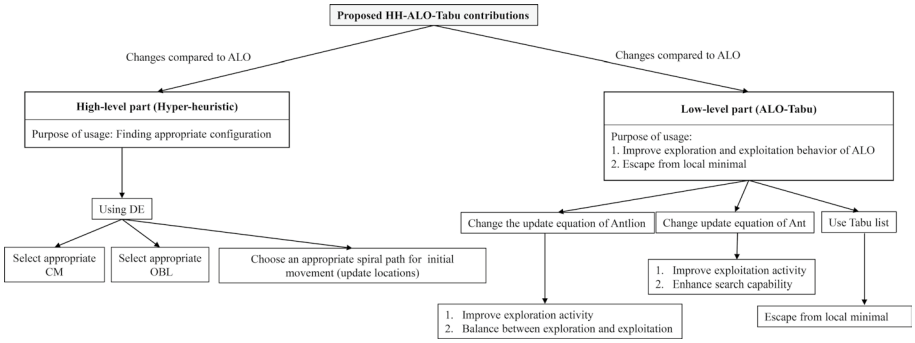

Fig. 2 The proposed HH-ALO-Tabu contributions

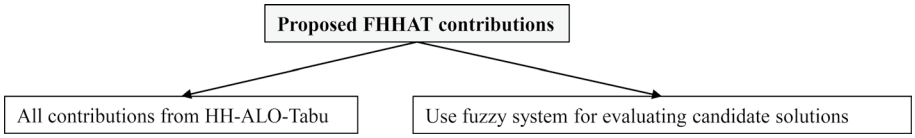

Fig. 3 The proposed FHHAT contributions

The remaining parts of this paper are summarized as follows. Section 2 illustrates some related works of data replication in the cloud. Section 3 explains the necessary background. In Sect. 4, the proposed hyper-heuristic ALO-Tabu (HH-ALO-Tabu) algorithm is introduced. Section 5 presents our new data replication algorithm (FHHAT). Section 6 discusses experiment results and Sect. 7 provides the main conclusions and some future research directions.

2 Related works

To guarantee high availability and low response times, it is essential to provide cloud users with low response times. Different data replication algorithms have been presented to enhance the service quality in the cloud environment. Gopinath and Sherly (Gopinath and Sherly 2018) proposed weighted data replication to improve cloud performance. The proposed algorithm calculated the replica factor according to the data popularity, current replication factor, and the number of active sites. It computes the data popularity based on the temporal locality which means a more recently requested file will be requested in the near future as well. In addition, it assigns a weight to each file according to the number of sites requested for the file. When the value of popularity is higher than the dynamic threshold, replication is performed. A large number of tasks can be handled more efficiently with the proposed algorithm based on the Hadoop distributed file system, HDFS (Borthakur 2007).

Xie et al. (Xie et al. 2017) introduced a new dynamic replication to improve data availability and reduce data movement. The authors considered three threshold factors for dependencies of files, file access frequencies, and storage of nodes. A dataset constraint is the dependency of files and the frequency of file access. In addition, the proposed algorithm limits data replication by determining the threshold value for storage capacity. Moreover, it categorizes files into three types fixed files, free-flexible files,

and constrained-flexible files. A fixed file cannot be copied due to the constraints of its attributes. A file is free-flexible for a site when the file size is lower than the available storage of the site and hence the file is freely replicated. Otherwise, the file is constrained-flexible. The proposed algorithm distributes the fixed files and flexible files and then file dependencies and access frequencies are computed among them. After that, the file for replicating needs to meet both file dependency and access frequency constraints. Using a case study, the authors showed how the algorithm could reduce data management costs.

An algorithm called multi-objective optimized replication management (MORM) is presented by Long et al. (Long et al. 2014). It looks at five factors: file unavailability, average service time, load, energy usage, and latency. MORM balances trade-offs among five objectives using an artificial immune algorithm (Luo et al. 2004). The proposed strategy builds individuals at random and then calculates each individual's fitness. Individuals with low fitness values are removed and then it repeatedly performs the mutation and selection operators to reproduce offspring. The experiments with CloudSim proved that MORM algorithm could decrease system unavailability and mean latency compared to the HDFS and MOE methods.

Hassan et al. (Hassan et al. 2009) proposed a MOE strategy for the data replication problems. MOE algorithm considers important parameters like access latency, availability, reliability, and storage costs. By avoiding connections with low bandwidth, it aims to minimize latency. There are conflicts among various objectives for example latency–reliability conflict. Latency increases as more copies are created and hence the propagation update cost is increased. Reliability tends to increase as more copies are stored since whenever a failure occurs, the system can continue with other replicas in the system. The proposed algorithm assumes solutions to the problem as chromosomes and searches for a near-optimal solution by the evolutionary computing paradigm. The proposed algorithm cloud provides a balance between the various conflicts, according to experiments.

To improve network utilization for cloud systems, Mansouri et al. (Mansouri et al. 2017) presented a DPRS. DPRS periodically computes data popularity according to users' access patterns and then uses the 80/20 concept to replicate frequently requested files. Three factors are taken into account when determining replica placement: the number of requests, storage capacity, and centrality. DPRS designs a parallel downloading method to reduce access time. It has been shown that DPRS can reduce response time and storage usage through comprehensive simulations.

PDR is an algorithm proposed by Mansouri and Javidi (Mansouri and Javidi xxxx) that reduces the number of communications between cloud systems. PDR algorithm tries to discover the correlated files by constructing the dependency matrix for files. Then it finds the most popular related files according to the mean number of accesses and replicates them. Whenever there is not enough storage to store a popular group of files, PDR replaces replicas with a fuzzy system based on multiple inputs (e.g., the number of accesses, replication cost, the last access time, and availability of file). According to CloudSim results, the PDR algorithm could reduce replication frequency and improve response time.

According to Chunlin et al. (Chunlin et al. 2019), SaaS applications can be replicated in the edge cloud. It provides load balancing of DNs, enhances network utilization, and reduces execution time. This problem is solved using a genetic algorithm that uses a fast non-dominated sorting process. Wu (Wu 2017) introduced a cost-effective replication algorithm for the cloud environment. By combining fitness function, selection procedure, mutation operator, and replacement population technique, the proposed method proposes a cost model for determining the number of replicas and their storage locations. There are three

types of costs associated with data management: storage costs, transfer costs, and computation costs.

Khalili Azimi (Khalili Azimi 2019b) developed a bee colony (beehive) based replication algorithm to reduce execution time in the cloud. The author considered the three-level hierarchical structure. There are two levels of networking: the first level has a cluster of personal computers connected via high-speed networks, and the second level has local area networks connected with lower-speed networks. The third level shows regions that have the lowest bandwidth. The proposed algorithm determines the suitable locations for new replicas based on the number of requests. So, the file will be stored on the site that has the most demand for that file.

Jayasree and Saravanan (Jayasree and Saravanan 2018) proposed a data replication based on particle swarm optimization (PSO) to increase the security of cloud data. The proposed algorithm divides files into some fragments and then the replicas of fragments stored in different locations based on the T-coloring technique and PSO algorithm. With a successful attack on a cloud node, the meaningful data is not released since no cloud node holds more than a single fragment.

We summarize the data replication algorithms based on different parameters in Tables 2 and 3. If the replication algorithm considers the specific parameter then we write the notation "Y" otherwise we write "N". Based on the summary in Tables 2 and 3, it is observed that data replication algorithms should find a solution by balancing the trade-offs among the several optimization objectives (e.g., response time, bandwidth consumption, load balancing, security, and financial costs). From analyzing the QoS factors, it is evident that response time is the most dominantly used objective. Some data replication algorithms try to utilize cloud resources such as bandwidth (Vulimiri et al. 2015) while others try to meet the service level agreement (SLA) without paying attention to the monetary impact of the replication decisions (Sousa and Machado 2012). Based on the results of the analysis, it is evident that each of the compared methods has its advantages and disadvantages in terms of the replication goals considered.

It appears that the majority of researchers are more concerned with replication efficiency and response time than execution budget. Therefore, the issue of data replication is still a challenge due to the heterogeneous and dynamic behavior of the cloud environment. In this paper, the proposed replication algorithm considers SA, system load variance (SLV), mean service time, and provider's expenditures (PEs). Since the mean service time and delay are low, the system is likely to be flexible and scalable, as well as complete tasks within the intended time frame. While satisfying the requirements of users, the proposed algorithm models the provider profit as monetary costs. A number of factors are taken into consideration when replicating the data, including response time, bandwidth consumption, load balancing, financial costs, meta-heuristics, fuzzy inference, replica placement, and replica replacement.

It becomes very difficult to find an optimal solution when there are more than three optimization objectives. Recently, researchers have contributed immensely by developing heuristics and meta-heuristics to help solve data replication. Since meta-heuristics can present better results than traditional methods by doing iterative processes in a reasonable time. Nevertheless, meta-heuristic algorithms sometimes have weaknesses (e.g. PSO suffers from slow convergence rate and weak local search). We can see in Tables 2 and 3, that most data replication algorithms use the common meta-heuristics and are not adequate for satisfying the needs of cloud computing. More importantly, adjusting the parameters of meta-heuristics is very challenging. To find the optimal configuration, we use the DE algorithm as a hyper-heuristic to improve the ALO algorithm.

Table 2 Dynamic data replication strategies for cloud computing

Strategy	Gopinath and Sherly (Gopinath and Sherly 2018)	Xie et al. (Xie et al. 2017)	Long et al. (Long et al. 2014)	Hassan et al. (Hassan et al. 2009)	Mansouri et al. (Mansouri et al. 2017)
Year	2018	2017	2014	2009	2017
Response time	Y	Y	Y	Y	Y
Bandwidth consumption	Y	N	Y	Y	Y
Load balancing	N	N	Y	N	N
Energy efficiency	N	N	Y	N	N
Heuristic technique	N	N	Y	Y	N
Fuzzy inference	N	N	Y	N	N
Security	N	N	N	Y	N
Replica placement	Y	Y	Y	Y	Y
Replica replacement	N	N	N	N	N
Replica selection	N	N	N	N	N
Parallel downloading	N	N	N	N	Y
Optimal number of replicas	Y	N	Y	Y	N
Evaluation environment	HDFS	Case study	CloudSim	Real environment	CloudSim
Main idea	Considering the popularity of file	Determining dataset dependencies	Using artificial immune algorithm	Considering latency, reliability and storage usage	Introducing a parallel downloading approach

Table 3 Dynamic data replication strategies for cloud computing

Strategy	Mansouri and Javidi (Mansouri and Javidi xxxx)	Chunlin et al. (Chunlin et al. 2019)	Wu (Wu 2017)	Khalili Azimi (Khalili Azimi 2019b)	Jayasree and Saravanan (Jayasree and Saravanan 2018)
Year	2018	2019	2017	2019	2018
Response time	Y	Y	Y	Y	Y
Bandwidth consumption	Y	Y	Y	Y	Y
Load balancing	N	Y	N	N	N
Energy efficiency	N	N	N	Y	N
Heuristic technique	N	Y	Y	Y	Y
Fuzzy inference	Y	N	N	N	N
Security	N	Y	N	N	Y
Financial costs	N	N	N	N	N
Replica placement	Y	Y	Y	Y	Y
Replica replacement	Y	N	N	Y	N
Replica selection	N	N	N	Y	N
Parallel downloading	N	N	N	N	N
Optimal number of replicas	N	Y	Y	N	N
Evaluation environment	CloudSim	Java	Real environment	MATLAB	–
Main idea	Considering file correlation	Use migration model	Define cost models	Consider three-level structure	Apply file fragmentation

Although the ALO algorithm has been successful in solving optimization problems, the random walking model and random initial population result in long execution times. In improved versions of the ALO, attempts have been made to replace the random walking distance for modeling ant's movement and improve population diversity. An opposition-based version of the ALO, named MALO, is proposed by Wang et al. (Wang et al. 2020) for speeding up convergence and preventing search deflation. Zawbaa et al. (Zawbaa et al. 2016) proposed an optimization method for solving the feature selection problem based on a "chaotic" version of the ALO technique. There is a random walk variable in ALO and when this variable linearly decreases, then in some cases ALO is trapped in local minima. The chaotic system improves the tradeoff between exploration and exploitation and removes this random walk variable. Emary et al. (Emary et al. 2016) proposed a binary version of ALO called BALO where each ant of ALO is updated based on the average of two positions that are determined by performing a random walk around the elite ant lion and a chosen ant lion. Using the BALO algorithm, two binary solutions are replaced by the average operator crossover.

Furthermore, meta-heuristic algorithms must be redesigned, their fitness functions defined, and initial population selection methods defined to enhance their quality. A fuzzy system is used in this work to evaluate fitness in the replication problem without focusing on related works. In Tables 2 and 3, it is evident that most researchers concentrate much more on replica placement than other main replication questions (e.g., replica replacement, the optimal number of replicas). Furthermore, the proposed method also considers replica placement along with replica replacement and replica number.

3 Background

3.1 Differential evolution (DE) algorithm

DE (Abd Elaziz and Mirjalili 2019) is an evolutionary computation (EC) method which uses three main operators (mutation, crossover, and selection). Firstly, DE randomly generates a population (i.e., PDE) to show *NS* solutions for the given problem as follows:

$$PDE_{ij} = L_j + rand_j \times \left(UD_j - LD_j\right), \quad j = 1,\ 2, \ldots, \dim_D, \quad i = 1,\ 2, \ldots, NS, \tag{1}$$

where $rand_j \in [0,\ 1]$ is a random number, LD_j and UD_j are the lower and the upper boundary for dimension j, and $\dim_D$ indicates the dimension for solution D. The best solution is determined based on the value of the objective function. Next, a new solution Z_i is created by the mutation operation as follows:

$$Z_i(t) = PDE_{r1}(t) + F \times \left(PDE_{r2}(t) - PDE_{r3}(t)\right), \tag{2}$$

where F is the mutation scaling factor and PDE_{rk}, $k = 1,\ 2,\ 3$ shows three solutions that are selected randomly from $[1 - NS]$. Now, it defines an offspring w_i based on the current parent solution PDE_i and Z_i as follows:

$$W_{ij}(t) = \begin{cases} Z_{ij}(t) & if\ \alpha_j \le CR\ or\ \delta_i, \\ PDE_{ij}(t) & otherwise, \end{cases} \tag{3}$$

where CR is the probability of crossover. The α_j is a random value from the dimension j and δ_i is a random index in the interval $[1 - \dim_D]$. Finally, the selection step is performed based on Eq. (4).

$$PDE_i(t+1) = \begin{cases} W_i(t) & if\ Fit\big(W_i(t)\big) \le Fit\big(PDE_i(t)\big), \\ PDE_i(t) & otherwise. \end{cases} \tag{4}$$

Iterations are repeated until the preset threshold is reached. Algorithm 1 represents DE's pseudo-code.

Algorithm 1. DE algorithm

Input: population size (NS), mutation factor (F), crossover probability (CR), Maximum iteration(T), dimension of the problem (dim)
Output: best solution (BS)
1 Initialize all NP solutions randomly where their positions are within the search space *//Based on Eq. (1)*
2 Evaluate all solutions and sort them in descending order based on their fitness
3 BS = first agent
4 t =1
5 While t <= T
6 {
7 For (i=1, i<= NP, i++)
8 {
9 Select three random solutions ($PDE_{r1}, PDE_{r2}, PDE_{r3}$) from the population and these solutions must differ from each other ($PDE_{r1} \neq PDE_{r2} \neq PDE_{r3}$)
10 $Z_i(t) = PDE_{r1}(t) + F \times (PDE_{r2}(t) - PDE_{r3}(t))$ *//Create a new solution with a mutation operator*
11 For (j=1, j<=dim, j++)
12 $W_{ij}(t) = \begin{cases} Z_{ij}(t) & if\ \alpha_j \le CR\ or\ \delta_i \\ PDE_{ij}(t) & otherwise \end{cases}$
13 If ($F(W_i) \le F(BS)$)
14 $BS = W_i$
15 }
16 t=t+1
17 } *//end while*
18 Return BS

3.2 Tabu search

The TS method (Mohammed and S.O, 2020) is a local search method that incorporates adaptive memory and responsive exploration. The implementation of the searching process becomes economically effective in terms of memory due to the adaptive memory feature of TS. Moreover, responsive exploration exploits good solutions through an intelligent search procedure while exploring new promising areas. Firstly, TS defines an initial solution and generates the neighbor solutions by modifying the available solution.

In every iteration, it uses the best new neighbor as its starting point, unless it is in the Tabu list. Tabu list saves the last obtained solutions and so the search process moves away from the recently selected solutions and escapes local optimal solutions. In each iteration, the Tabu list is updated and the best solution is separately kept. Algorithm 2 represents the step of the TS method.

Algorithm 2. TS algorithm

Input: population size (NS), mutation factor (F), crossover probability (CR), Maximum iteration(T), dimension of the problem (dim)
Output: best solution (BS)
1 Generate a random initial solution (RS)
2 Initialize TL
3 BS = TS
4 t=1
5 While t <= T
6 {
7 New_RS = Update (RS) //update current solution with different movement
8 Evaluate the New_RS
9 If (F(New_RS) ≤ BS && (New_RS) not in TL)
10 BS=New_RS; RS=New_RS
11 Else
12 Update TL
13 t=t+1
14 }
15 Return BS

3.3 ALO algorithm

Using the ALO, Mirjalili (Mirjalili 2015) proposed a bio-inspired method based on the behavior of ant lions in nature that hunt ants. During the experiment, they considered five main phases: a random walk of ants, the creation of traps by ant lions, the entrapment of ants in traps, the capture of ants by ant lions, and the re-creation of traps. Mirjalili (Mirjalili 2015) compared the ALO algorithm with some other intelligent algorithms such as PSO, GA, and Cuckoo system and tested it on some mathematical functions and engineering problems (e.g., the design of ship propeller) to demonstrate that the ALO algorithm has better characteristics in convergence speed, accuracy, local optima avoidance, and robustness.

Using ALO, this paper finds an optimal solution for the objective functions. In the standard ALO method, search agents are ants that move over the decision space. Ant lions can hunt them by building a trap. Figure 4 indicates the ant lion hunting behavior. By roulette wheel process, each ant's position is updated according to the ant lion chosen by each iteration (i.e., *Elite*). The RWS process assigns a higher selection probability to a solution with the better fitness function as the ant lion with a larger trap and so can prey on more ants. ALO strategy can be explained in the following subsections.

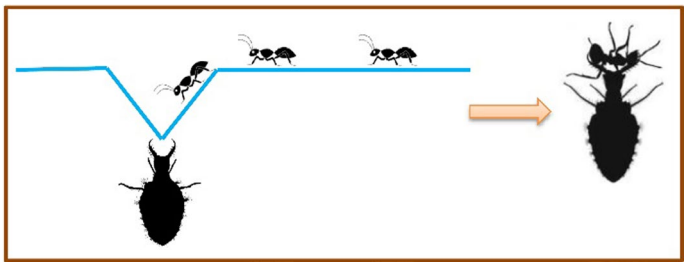

Fig. 4 Behavior of an ant lion hunting

3.3.1 Ants' positions and movements

Ants move around the search space based on a random walk as follows (Mirjalili 2015).

$$X(A_i^t) = [0,\ CumSum\,(2s(t_k) - 1), \ldots, CumSum(2s(t_T) - 1)]; \quad k = 1,\ 2, \ldots, T, \tag{5}$$

where *CumSum* computes the cumulative sum of walks, t indicates the step of random walk, T is the max iteration, and $s(t_k)$ is a stochastic function which is expressed as Eq. (6).

$$s(t_k) = \begin{cases} 1 & rand > 0.5, \\ 0 & rand \leq 0.5. \end{cases} \tag{6}$$

In each iteration, the random walk may accomplish slowly or very fast. Figure 5 represents an example of this random walk in the first four iterations. We can see that ants are walking very fast.

When the ants are walking toward the antlion, they may walk slowly or very quickly. Figure 5 shows an example of this random walk in its first four iterations. Using the random walk, ants' movement speed can be scaled. It can be seen that a random walk is applied in Eq. (11), where ants are moved toward antlions in traps with small steps in the early iterations and with larger steps in the last iterations.

The ants' positions are kept in the following matrix.

$$P_{ant} = \begin{bmatrix} A_{1,1} & A_{1,2} & \ldots & A_{1,n} \\ A_{2,1} & A_{2,2} & \ldots & A_{2,1} \\ \ldots & & & \\ A_{n,1} & A_{n,2} & \ldots & A_{n,m} \end{bmatrix}, \tag{7}$$

where P_{ant} is a matrix of ants' positions, $A_{i,j}$ indicates the jth decision variable of the ith ant, n and m show the number of ants and number of variables, respectively.

$$P_{antlion} = \begin{bmatrix} Al_{1,1} & Al_{1,2} & \ldots & Al_{1,n} \\ Al_{2,1} & Al_{2,2} & \ldots & Al_{2,1} \\ \ldots & & & \\ Al_{n,1} & Al_{n,2} & \ldots & Al_{n,m} \end{bmatrix}, \tag{8}$$

where $P_{antlion}$ is matrix of ant lions' positions, and $Al_{i,j}$ indicates the jth decision variable of the ith ant lion, n and m show the number of ant lions and number of variables, respectively.

Iteration 2 $[0 \quad cumsum\,(2\times([0.18] > 0.5) - 1] = [0 \quad cumsum\,((2\times 0) - 1] = cumsum\,([0 \quad -1]) = [0 \quad -1]$

Iteration 3 $[0 \quad cumsum\,(2\times([0.18] > 0.5, [0.74] > 0.5) - 1] =$
$[0 \quad cumsum\,((2\times 0) - 1, (2\times 1) - 1)] = cumsum\,([0 \quad -1 \quad +1]) = [0 \quad -1 \quad 0]$

Iteration 4 $[0 \quad cumsum\,(2\times([0.18] > 0.5, [0.74] > 0.5, [0.65] > 0.5) - 1] =$
$[0 \quad cumsum\,((2\times 0) - 1, (2\times 1) - 1, (2\times 1) - 1) = cumsum\,([0 \quad -1 \quad +1 \quad +1]) = [0 \quad -1 \quad 0 \quad +1]$

Fig. 5 An example of the cumulative sum

Fitness values for ants and ant lions are obtained during optimization and stored in F_{ant} and $F_{antlion}$, respectively.

$$F_{ant} = \begin{bmatrix} f([A_{1,1}\ A_{1,2}\ \ldots\ A_{1,n}]) \\ f([A_{2,1}\ A_{2,2}\ \ldots\ A_{2,1}]) \\ \ldots \\ f([A_{n,1}\ A_{n,2}\ \ldots\ A_{n,m}]) \end{bmatrix}, \tag{9}$$

$$F_{antlion} = \begin{bmatrix} f([Al_{1,1}\ Al_{1,2}\ \ldots\ Al_{1,n}]) \\ f([Al_{2,1}\ Al_{2,2}\ \ldots\ Al_{2,1}]) \\ \ldots \\ f([Al_{n,1}\ Al_{n,2}\ \ldots\ Al_{n,m}]) \end{bmatrix}. \tag{10}$$

Ants move and update their positions based on the random walk in each iteration and so to keep the movements of ants within the decision space, the random walks are normalized based on Eq. (11).

$$X(A_{i,j}^t) = \frac{(X(A_{i,j}^t) - \alpha_{i,j}) \times (d_j - c_j)}{(\beta_{i,j} - \alpha_{i,j})} + c_j, \tag{11}$$

where $\alpha_{i,j}$ and $\beta_{i,j}$ indicate the minimum and maximum of random walk for the ith ant in the jth dimension [the random walk is obtained by Eq. (5)], c_j'' and d_j'' respectively represent the lower bound and upper bound in the jth dimension at the tth iteration after the ant walks around the selected ant lion [with Eqs. (12) and (13)].

3.3.2 Entrapping ants inside pits

Since ant lion traps affect ant random walks, the boundaries of ant random walks are adapted in each step so that the ant moves within a hypersphere around the chosen ant lion. Equations (12) and (13) mathematically model this behavior as follows (Mirjalili 2015).

$$c_j' = Al_{sj}^t + c_j^t, \tag{12}$$

$$d_j' = Al_{sj}^t + d_j^t, \tag{13}$$

where Al_{sj}^t indicates the jth dimension of the selected ant lion at the tth iteration. c_j^t and d_j^t show the value of the lower bound and the value of the upper bound in the jth dimension at the tth iteration after reducing the boundaries with Eqs. (12) and (13), respectively. c_j' and d_j' indicate the random walk around selected ant lion at the lower bound and the upper bound, respectively.

3.3.3 Building trap

A RWS method is used to model the hunting capability of ant lions. This method assigns a high chance to the fittest ant lion (i.e., the ant lion with the highest fitness in maximization

problem and the lowest one in minimization problem) for hunting ants. In each iteration, each ant can fall into only one trap of ant lions.

The process of ant lion selection with the RWS method can be seen in Fig. 6. It can be seen that we have an ant lion population with 10 ($P1$, $P2$,…,$P10$) members with different fitness values. Since the problem is a minimization problem, the ant lion with lower fitness is a better and fitter one. To facilitate the construction of the roulette wheel, all fitness values are recalculated by Eq. (14) so that the lower one gets a higher value and then the probability of selection is calculated by Eq. (15).

$$new\_fitness_i = \frac{1}{f(Al_{P_i})} \quad i = 1,\ 2,\ 3, \ldots, 10, \tag{14}$$

$$probability\_selection_i = \frac{new\_fitness_i}{\sum_{i=1}^{10} new\_fitness_i}. \tag{15}$$

The roulette wheel is based on the selection probability of ant lions, so the ant lion with the lowest fitness has a higher chance of being selected. A random value between $r(0, 1)$ is generated to select the ant lion. In Fig. 6, r is equal to 0.44 and so the first ant lion is selected.

3.3.4 Sliding ants towards ant lion

Based on the selection probability of ant lions, the roulette wheel favors the ant lion with the lowest fitness center of the hole. This behavior can be modeled by decreasing the radius of the ant's random walk as indicated in Eqs. (16) and (17).

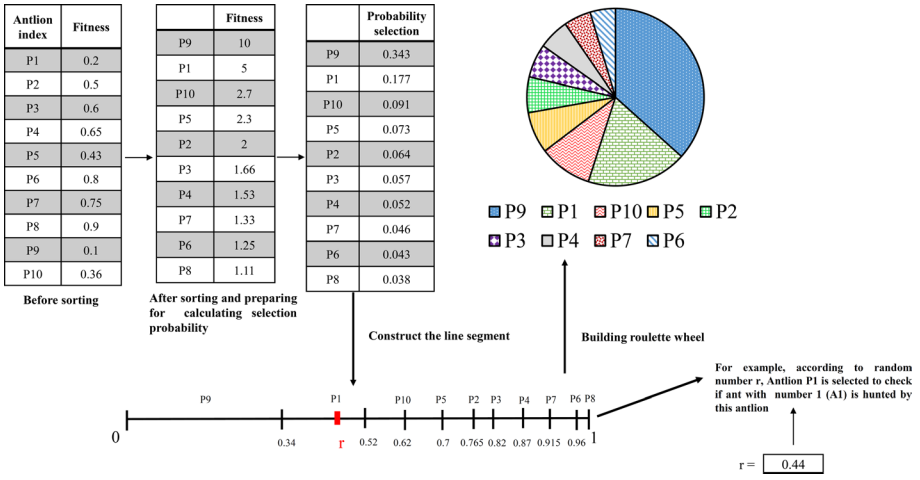

Fig. 6 Ant lion selection process

$$c^t = \frac{c^t}{R}, \tag{16}$$

$$d^t = \frac{d^t}{R}. \tag{17}$$

R is a ratio and can be determined in Eq. (18).

$$R = 10^{\lambda}\frac{t}{T}, \tag{18}$$

where t and T show the current iteration and the max iteration, respectively. λ is a constant that is adjusted according to the iteration number and is given by Mirjalili (2015):

$$\lambda = \begin{cases} 2 & t > 0.1T, \\ 3 & t > 0.5T, \\ 4 & t > 0.75T, \\ 5 & t > 0.9T, \\ 6 & t > 0.95T. \end{cases} \tag{19}$$

3.3.5 Catching prey and reconstructing the pit

Whenever an ant falls into the pit, its jaws are caught by the ant lion. According to the ALO approach, ant lions catch prey when their corresponding ants become fitter. Now, the ant lion changes its position to the position of the hunted ant, and hence its ability for hunting new ants increases. This assumption can be mathematically considered based on Eq. (20).

$$Al_i^t = \left\{ A_i^t \quad if\ f(A_i^t) < f(A_i^t), \right. \tag{20}$$

where t is the current iteration. Al_j^t and A_j^t indicate the position of the jth ant lion and the jth ant for the tth iteration, respectively.

3.3.6 Elitism

An important characteristic of evolutionary techniques is an elitism that maintains the best solution throughout the optimization process. ALO method saves the best ant lion in each iteration as the elite ant lion and then considers the mean of the elite ant lion and the chosen ant lion by using the selection method as follows (Mirjalili 2015).

$$A_i^{t+1} = \frac{X(Al_s^t) + X(Al_{Elite}^t)}{2}, \tag{21}$$

where A_i^{t+1} shows the position of the ith ant in the $t+1$th iteration. $X\left(Al_s^t\right)$ and $X\left(Al_{Elite}^t\right)$ indicate the random walk around the selected ant lion and random walk around the elite ant lion at the tth iteration, respectively. Algorithm 3 shows the pseudo-code of the ALO algorithm.

Every ant has two random walks that determine its new position. First, walk around a randomly selected ant lion. Secondly, walks around elite antlion (i.e., the ant lion with the best fitness). The level of reduction from upper and lower boundaries for the ants is calculated based on Eqs. (16)–(19). In this regard, the level of reduction from upper and lower boundaries for the ants is calculated based on Eqs. (16)–(19).

Considering the Schwefel function $f(x) = 418.9829 \times 10 - \sum_{i=1}^{10} x_i \times \sin\left(\sqrt{|x_i|}\right)$ and a problem with 10 dimensions, we want to apply Eqs. (11)–(13) and (16)–(19) from standard ALO simultaneously on the first ant in iteration 51. As shown in Fig. 7, the first ant's original boundaries and the level of reduction from these boundaries are shown. The lower and upper bounds for this function arE−10 and 10, respectively. The level of reduction that is obtained by Eqs. (16)–(19) is −0.8929 for the lower bound and 0.8929 for the upper bound.

In Fig. 8, the random walk of the ant around the ant lion is shown according to Eqs. (12) and (13).

Figure 9 shows the impact of normalization based on Eq. (11) on a random walk.

The updating position of the ant, besides the random walk around the random ant lion ($X(Al_s^t)$) is related to the random walk around the elite ant lion ($X(Al_{Elite}^t)$). Figure 10a shows the random walk of the ant around the elite and randomly selected ant lion. In addition, the previous position and the updated position of the ant can be seen in Fig. 10b.

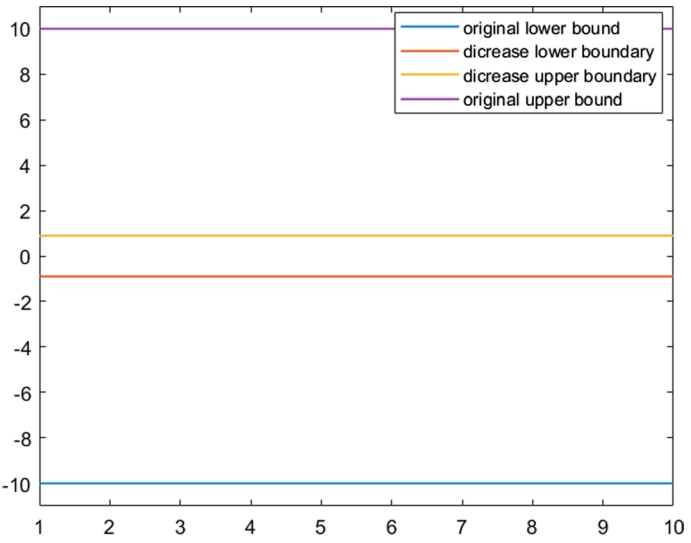

Fig. 7 Schwefel function boundary description

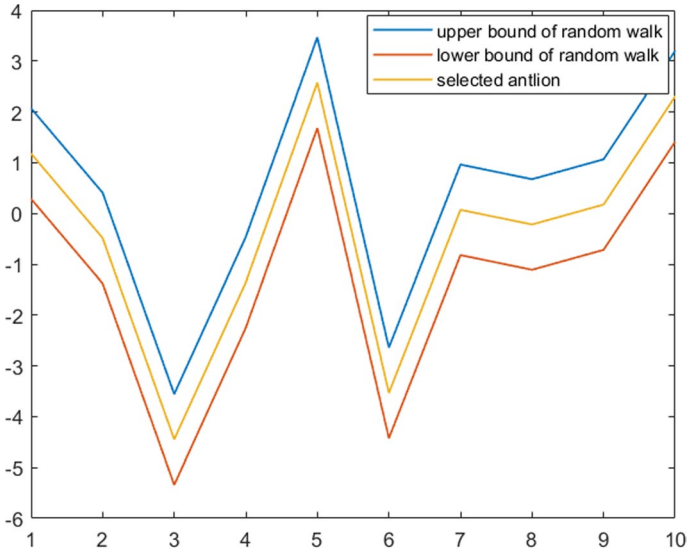

Fig. 8 Random walk around the selected ant lion

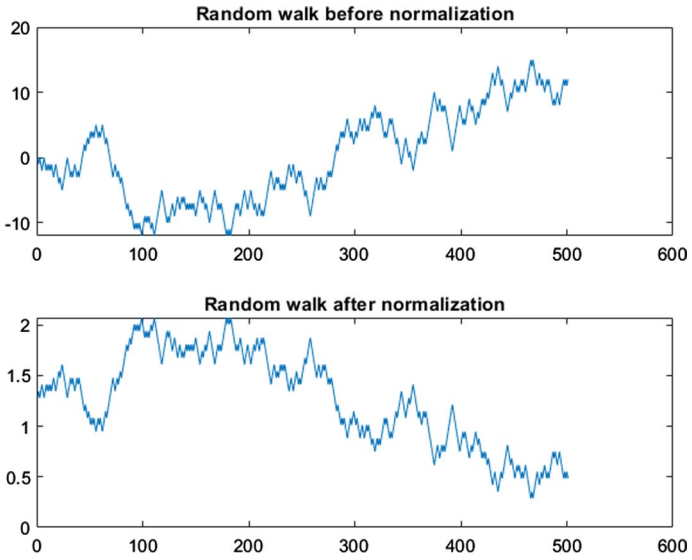

Fig. 9 Random walk before and after normalization

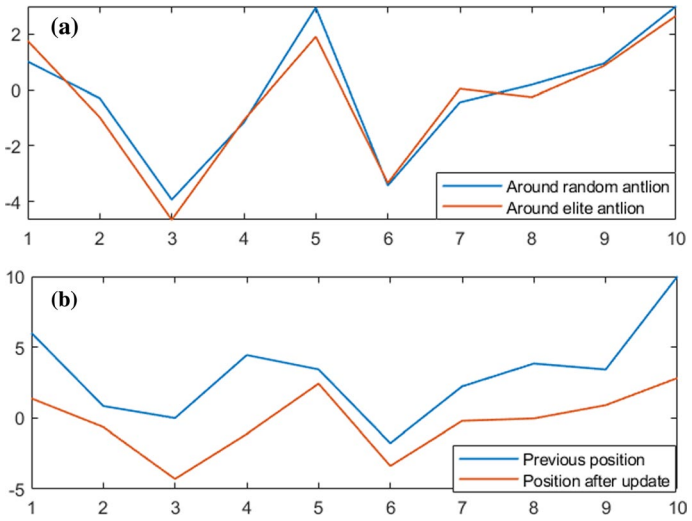

Fig. 10 **a** Random walks around random and elite ant lion, and **b** ant's position before and after updating

Algorithm 3. ALO algorithm

```
Initial the population of ants and ant lions randomly
Compute the values of fitness for initial ants and ant lions
Find the best ant lion as the elite based on the fitness value
While (Num_iteration <= Max_iteration)
{
  For all ants do
    {
      Choose an ant lion by using the roulette wheel process
      Update the lower and upper bounds based on Eq. (16) and Eq. (17)
      Generate two random walks around the elite and the roulette chose ant lion
      Normalize them based on Eq. (11)
      Update the ants' positions based on Eq. (21)
    } //end for
  Compute the values of fitness for all ants
  Replace the ant lion with its corresponding ant if it is better //based on Eq. (20)
  If an ant lion's fitness value is better than the elite then update the elite
} // end while
Return the elite
```

3.4 Chaotic map (CM)

Based on a variety of probability distributions (e.g., Gaussian and uniform), the swarm algorithms define a random population. Nevertheless, the randomness of such processes may cause slow convergence and reduce the solution quality. CMs are often used as a solution to this problem to generate a search space with a reasonable level of diversity (Santos

and Mariani 2008). A set of chaotic ALO versions are introduced based on non-invertible maps to use the advantages of CMs. Ten CMs are listed in Table 4.

In meta-heuristic algorithms, one of the main steps is to generate new solutions that can improve previous ones. It must search for important areas around the global optimum and escape from any local optimum. To reduce computing costs and enrich the searching behavior, a good balance between exploration and exploitation is necessary. If the algorithm has proper exploration but poor exploitation then it is successful in escaping from the local solution but it is failing in convergence to the global solution. On the other hand, an optimization algorithm with good exploitation and poor exploration has a low chance of discovering global optimum in problems with a lot of local minima. Many studies discussed the exploration and exploitation tradeoffs. They indicated that chaos is one of the cheapest techniques recently used to boost the performance of optimization methods (Gandomi et al. 2013b; Coelho and Mariani 2012). Chaotic systems help to avoid premature convergence. Since nature-inspired optimization techniques aren't always effective, uniform and Gaussian distributions can be used to achieve randomness.

Unlike randomness, chaos has dynamic and statistical characteristics that make variables pass through all states without repeating. In other words, a chaotic algorithm is a special kind of random-based technique that employs chaotic variables rather than random variables. The stochastic search based on probabilities is, therefore, slower while downright searches are faster.

Many applications of chaos theory have successfully been applied to enhance performance and develop nonlinear dynamics, including secure transmission (Suneel 2006), DNA computing (Arena et al. 2000), and image processing (Manganaro and Gyvez 1997). Furthermore, several works have modified meta-heuristic optimization methods with chaos theory (Alatas et al. 2009; Coelho and Mariani 2008) to improve global convergence and prevent local solutions from being trapped.

3.5 Opposition-based learning (OBL)

As part of the OBL technique, an individual's fitness in a current position is compared with that of an individual placed in the opposite direction. Then, the position with fitter fitness is chosen for the next generations (Tizhoosh 2005). Consider the first dimension of the first agent position in the ant population $A_{1,1} \in [LW, UW]$ and so the opposite number $\overline{A_{1,1}}$ is obtained based on Eq. (22):

$$\overline{A_{1,1}} = UW + LW - A_{1,1}, \tag{22}$$

where LW and UW indicate the lower and the upper boundaries in the problem, respectively. We can extend this definition for the jth dimension of the ith solution $\overline{A_i} \in R^{\dim}$ as follows:

$$\overline{A_{i,j}} = UW_j + LW_j - A_{i,j} \quad j = 1, 2, \ldots, \dim, \tag{23}$$

If $F(\overline{A_i})$ is better $F(A_i)$ then $\overline{A_i}$ is saved. Otherwise, A is selected. Therefore, the population shows only the best solutions from $PAnt$ and $\overline{PAnt}$ according to the OBL technique.

A variety of OBL algorithms have been developed, such as quasi-OBL, QOBL (Rahnamayan et al. 2007), quasi-reflection OBL, QROBL (Ergezer et al. 2009), super-opposite based learning, SOBL (Hassan et al. 2009), and reflected extended opposition, REO (Kaucic 2013). The description of these algorithms are explained as follows:

Table 4 Description of the chaotic maps (Abd Elaziz and Mirjalili 2019)

Number	Map	Formula	Condition
1	Chebyshev	$x_{k+1} = \cos(k \cos^{-1}(x_k))$	–
2	Circle	$x_{k+1} = x_k + b - \left(\frac{a}{2\pi}\right) \sin(2\pi x_k) \bmod$	A = 0.5 and b = 0.2, it creates chaotic in (0, 1)
3	Gauss/Mouse	$x_{k+1} = \begin{cases} 1 & x_k = 0 \\ \frac{1}{\bmod(x_k,\ 1)} & Otherwise \end{cases}$ $where: \ \frac{1}{\bmod(x_k,\ 1)} = \frac{1}{x_k} - \left[\frac{1}{x_k}\right]$	Creates chaotic in (0, 1)
4	Iterative	$x_{k+1} = \sin\left(\frac{a\pi}{x_k}\right)$	$a \in (0,\ 1)$ is an appropriate parameter
5	Logistic	$x_{k+1} = a x_k (1 - x_k)$	x_k is the kth chaotic number, and $x_0 \in (0,\ 1)$, a = 4
6	Piecewise	$x_{k+1} = \begin{cases} \frac{x_k}{p} & 0 \le x_k < p \\ \frac{x_k - p}{0.5 - p} & p \le x_k < 0.5 \\ \frac{1 - p - x_k}{0.5 - p} & 0.5 \le x_k < 1 - p \\ \frac{1 - x_k}{p} & 1 - p \le x_k < 1 \end{cases}$	$x_0 \in (0,\ 0.5)$ is the control parameter and $x \in (0,\ 1)$ and $p \neq 0$
6	Sine	$x_{k+1} = \frac{a}{4} \sin(\pi x_k)$	$0 < a < 4$
8	Singer	$x_{k+1} = \mu(7.86 x_k - 23.31 x_k^2 + 28.75 x_k^3 - 13.3 x_k^4)$	$\mu \in (0.9,\ 1.08)$
9	Sinusoidal	$x_{k+1} = a x_k^2 \sin(\pi x_k)$	a = 2.3
10	Tent	$x_{k+1} = \begin{cases} \frac{x_k}{0.7} & x_k < 0.7 \\ \frac{10}{3}(1 - x_k) & x_k \ge 0.7 \end{cases}$	–

(1) The quasi-opposite ($\overline{A_q}$) of agent A is obtained based on Eq. (24), which is a uniform random solution between the middle point (Mid) and the opposite solution $\overline{A}$.

$$\overline{A_q} = Mid + (Mid - \overline{A}) \times rand. \tag{24}$$

(2) The quasi-reflected solution ($\overline{A_{qr}}$) of agent A is obtained based on Eq. (25), which is a random solution between the middle point $Mid = (UW + LW)/2$ and agent A.

$$\overline{A_{qr}} = UW + (Mid - A) \times rand. \tag{25}$$

(3) The super-opposite solution ($\overline{A_{so}}$) of agent A is obtained based on Eq. (26), which is a uniform distribution.

$$\overline{A_{so}} = \begin{cases} \overline{A} + (UW - \overline{A}) \times rand, & if\ \overline{A} > Mid, \\ LW + (\overline{A} - LW) \times rand, & otherwise. \end{cases} \tag{26}$$

(4) REO ($\overline{A_{reo}}$) reflects the extended opposite point to obtain the reflected extended opposite point that is defined by Eq. (27).

$$\overline{A_{reo}} = \begin{cases} rand(A_j,\ LW_j),\ if\ A_j > \frac{(UW_j + LW_j)}{2}, \\ rand(UW_j,\ A_j),\ if\ A_j < \frac{(UW_j + LW_j)}{2}. \end{cases} \tag{27}$$

3.6 Fuzzy system

Fuzzy systems have been widely used to solve decision-making problems because of their understandable and straightforward structure. It often presents suitable decisions based on inputs and knowledge-based rules where there is no reliable strategy exists. A fuzzy system consists of three main steps (1) fuzzification, (2) rule evaluation and (3) defuzzification. Figure 11 indicates the three stages along with the required parameters.

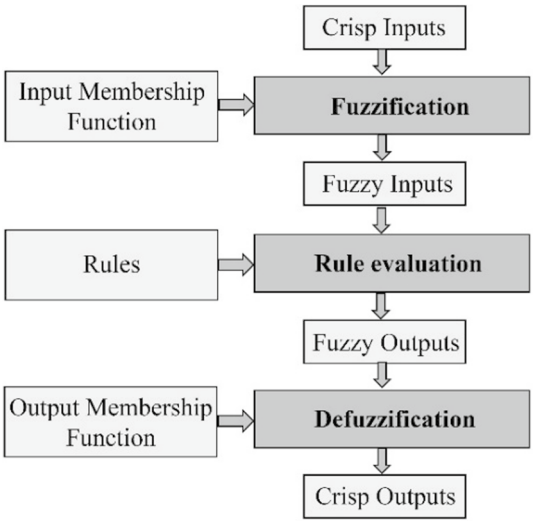

Fig. 11 Steps of fuzzy logic

- *Fuzzification* a membership function converts crisp quantities into fuzzy quantities.
- *Rule evaluation* the fuzzy inputs are applied to fuzzy rules. Then, the membership functions of all rule consequents are grouped into a single fuzzy set.
- *Defuzzification* it transforms the fuzzy output into crisp values.
- In most cases, humans or machines perform crisp actions or make crisp decisions, so fuzzy results should be converted. Thus, defuzzification produces a single number and an aggregate set. Centroid defuzzification is the most popular defuzzification method.

4 Proposed hyper-heuristic ALO-Tabu (HH-ALO-Tabu) algorithm

As a result of the proposed algorithm (HH-ALO-Tabu), the performance of the standard ALO is improved and the disadvantages of the standard ALO are removed, and the algorithm achieves two main goals as follows:

(1) By analyzing the CM, the strategy of the OBL, the initial spiral path, and the population ratio, the optimal configuration is determined.
(2) It enhances the exploration and exploitation capability of ALO in two ways:

(2.1) improve the ant's random walk with TS,
(2.2) improve the search process of ant lions with a novel relocation process.

Two types of improvements are performed to achieve these goals. For the first goal, the DE strategy is used to find the best configuration from different chaotic functions, a set of spiral movements. The OBL strategies are considered for generating the appropriate initial population and the initial spiral path. The proposed algorithm is called a hyper-heuristic algorithm since a heuristic (DE) algorithm is applied to determine the best (i.e., optimal) configuration for the ALO-Tabu algorithm. For the second goal, four different random walks are presented for ants, and selecting the initial one is determined by DE. Furthermore, a TS concept determines the order of choosing them.

Next, an elite ant lion relocation method based on iterations, ants, and ant lions' fitness is presented. The following subsections provide details about HH-ALO-Tabu's algorithm. Figure 12 illustrates the proposed method's main framework.

Two levels of hyper-heuristic techniques exist (low- and high-level). A local search approach at the low level is implemented and runs quickly. The high level contains the controller of low-level approaches (Ergezer et al. 2009) to converge the optimal solutions. An optimal configuration for ALO-Tabu is found by using a hyper-heuristic based on the DE algorithm (Sousa and Machado 2012). These subsections describe ALO-TAbu and HH-ALO-Tabu in more detail.

4.1 ALO-Tabu

For example, the ALO algorithm has a long execution time, premature convergence due to lack of exploration and exploitation capabilities, and is stuck at the local optima for some problems. Therefore, it is necessary to improve the search capability. In this paper, this improvement is occurred by a new ant's random walk model and a novel ant lion's relocation. The ant's random walking approach considers the maximum iteration number to determine the route of the ant's random walking. Therefore, this mechanism has a negative

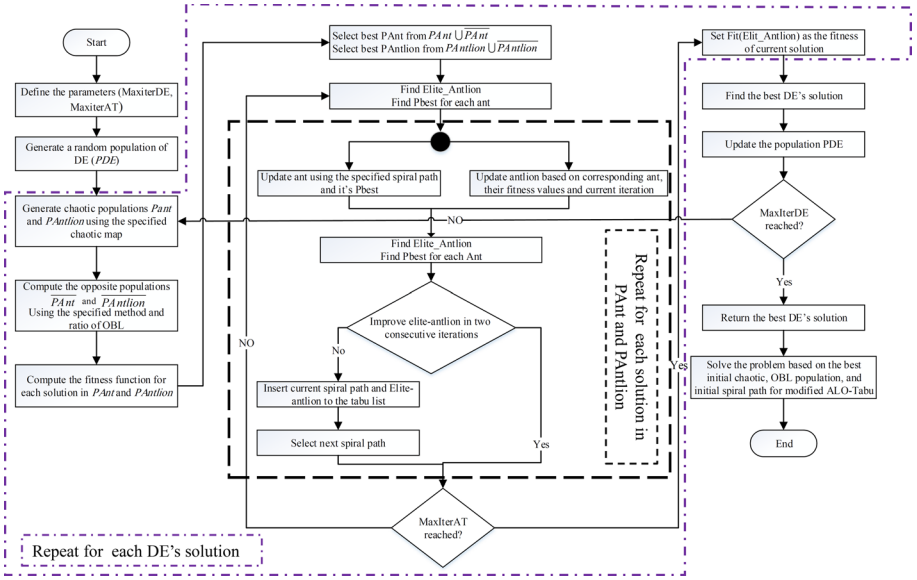

Fig. 12 The proposed HH-ALO-Tabu framework

effect on the execution time of the original ALO algorithm. In the ALO algorithm, some ant lions cannot be updated by ant due to the rand walk of ant (i.e., food shortage appears for the ant lions). Consequently, several innovations to the ALO algorithm are performed to balance the random walk size (i.e., search step size) and the movement of ant lions (i.e., ant lions' relocation) (Dong et al. 2021; Saha and Mukherjee 2018; Pradhan et al. 2018; Kılıç and Yüzgeç 2019).

4.1.1 Improve ants' random walk

With the roulette technique, the global search capability is improved by random walks of ants around the elite. Nevertheless, there are some problems as follows:

(1) As iterations increase, the diversity of the population gradually declines, resulting in a premature phenomenon.
(2) Whenever the elite and roulette ant lion are not in optimal locations, the convergence speed increases.

To provide more diversity in ant populations (i.e., exploration ability), we consider four spirals as mathematical models.

Archimedes spiral curve (*AR*) it is a constant velocity spiral that is presented by Eq. (28). Figure 13 indicates the two-dimensional Archimedes spiral.

$$\begin{cases} x = a + b.l\cos(n\pi l), \\ y = a + b.l\sin(n\pi l), \end{cases} \tag{28}$$

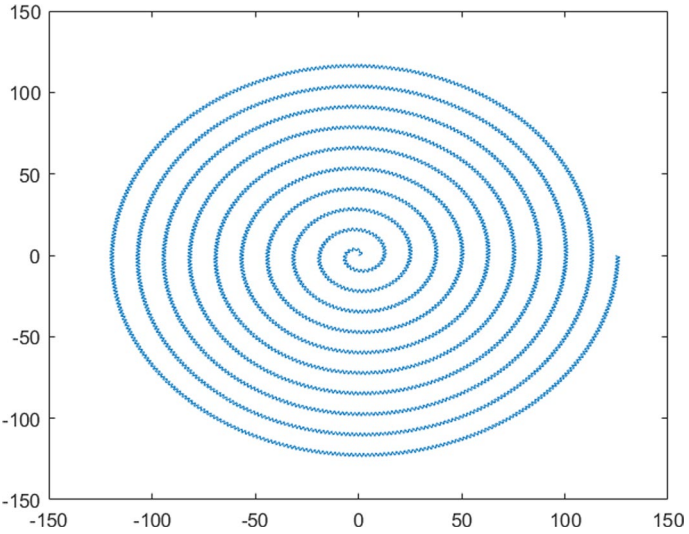

Fig. 13 Two-dimensional Archimedes spiral

where a and b are two constant. The a parameter indicates the start location coordinates, b determines the distance of turns from each other, and n represents the number of turns in the spiral.

Figure 14 shows various Archimedes spirals, their differences are in different values set for the control parameters (a, b, n, and l).

Logarithmic spiral curve (*LO*) it is an equiangular spiral that is presented by Eq. (29). Figure 15 indicates the two-dimensional of the logarithmic spiral.

$$\begin{cases} x = a.e^{l}\cos(n\pi l), \\ y = a.e^{l}\sin(n\pi l), \end{cases} \tag{29}$$

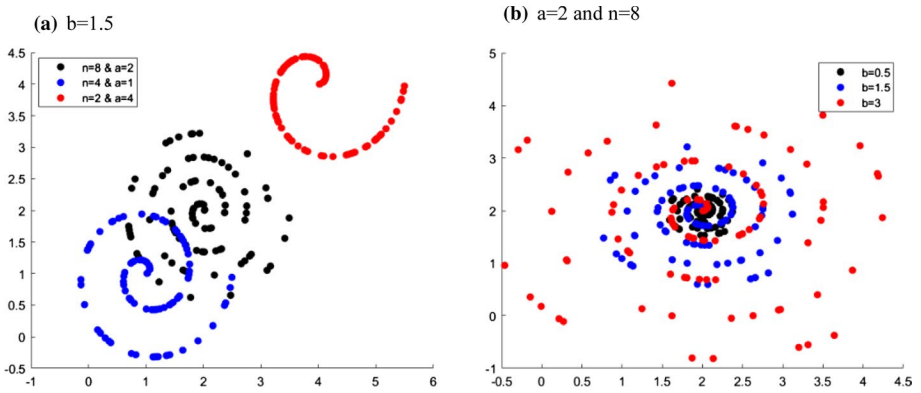

Fig. 14 Different Archimedes spirals with various values

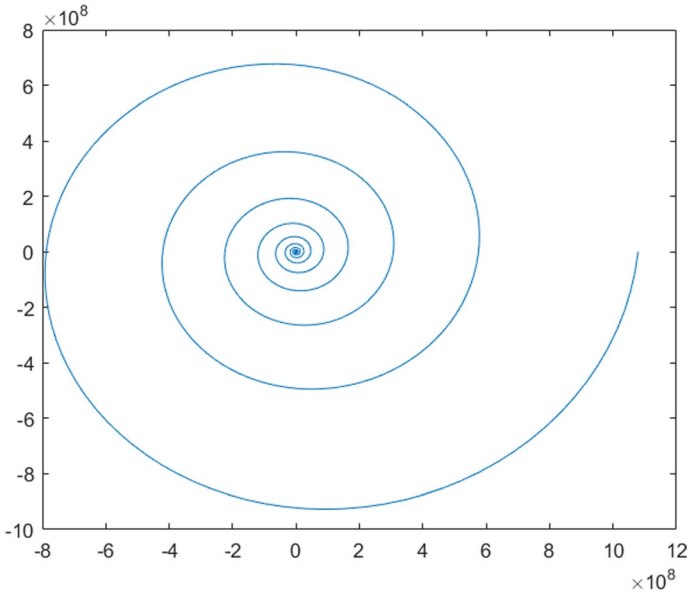

Fig. 15 Two-dimensional logarithmic spiral

where n indicates the number of turns in the spiral and l is a constant to control the spiral shape.

Figure 16 shows various logarithmic spirals. Their differences are in different values set for the control parameters (i.e., a, n, and l).

Rose spiral curve (*RO*) it consists of parameter a which controls the length of the leaves, and parameter n which controls the leaves' number, size, and period. The rose spiral is defined by Eq. (30). The two-dimensional image of the rose spiral is shown in Fig. 17.

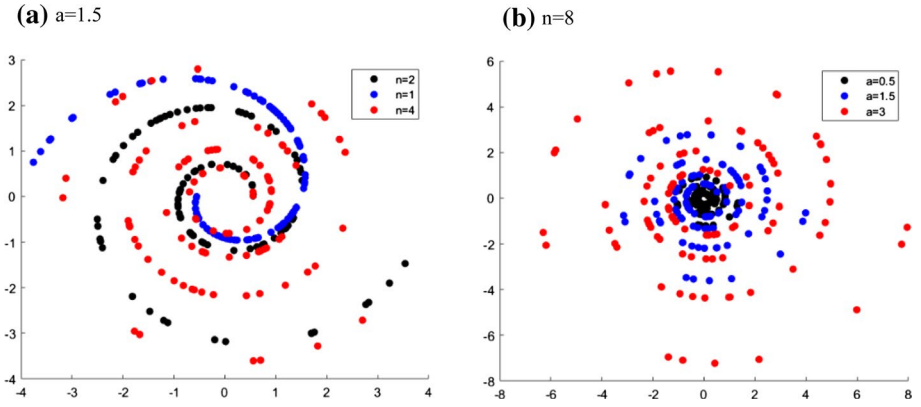

Fig. 16 Different logarithmic spirals with various values

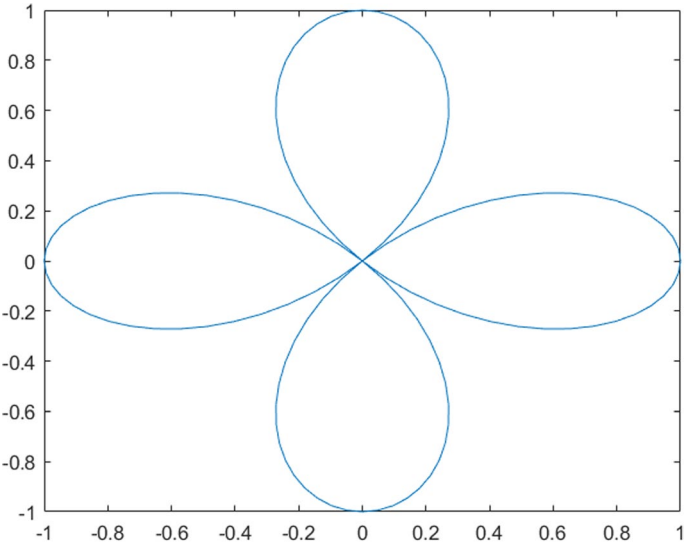

Fig. 17 Two-dimensional rose spiral

$$\begin{cases} x = a.\cos(n\pi l)\sin(l), \\ y = a.\sin(n\pi l)\cos(l). \end{cases} \tag{30}$$

Hypotrochoid (*HY*) *HY* follows a point moving inside a fixed circle with radius a, attached to a circle with radius b. The parameter c shows the distance between this point to the center of the inner circle. HY is defined by Eq. (31). Figure 18 indicates the two-dimensional curve of HY.

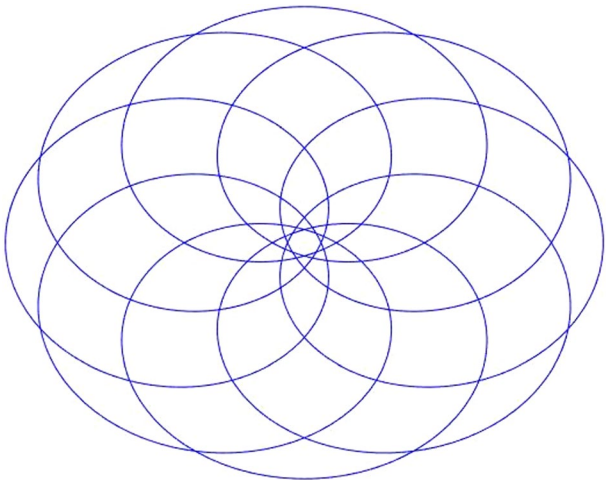

Fig. 18 Two-dimensional hypotrochoid

$$\begin{cases} x = (a-b).\cos(n\pi l) + c.\cos\left(\dfrac{a-b}{b} n\pi l\right), \\ y = (a-b).\sin(n\pi l) + c.\sin\left(\dfrac{a-b}{b} n\pi l\right). \end{cases} \tag{31}$$

According to ALO standards, ants are updated by searching the neighborhood of a random ant lion whose location is determined by the roulette wheel and *Elite*. While we adapt the spiral paths to update ants and simulate their movement of them in an ant lion's trap. In the proposed ant's random walk, the movement of an ant for the $T+1$th iteration is toward the *Elite* ant lion (from Tth iteration) and its best position which it's experienced so far (from iteration 1 to T). For example, the logarithmic spiral path is explained in the following formula to indicate how the position of ants is updated.

$$R_{A^t_{pbest}} = D_1.e^{bl}.\sin(2\pi l), \tag{32}$$

$$R_{Al^t_{Elie}} = D_2.e^{bl}.\cos(2\pi l), \tag{33}$$

$$A_i^{t+1} = \frac{R_{A^t_{pbest}} + R_{Al^t_{Elite}}}{2}, \tag{34}$$

$$D_1 = \left|A_i^t + \left(A^t_{pbest} - A_i^t\right)\right|, \tag{35}$$

$$D_2 = \left|A_i^t + \left(Al^t_{Elite} - A_i^t\right)\right|, \tag{36}$$

where $R_{A^t_{pbest}}$ and $R_{Al^t_{Elite}}$ denote the random walk around the personal best of ith ant and *Elite* ant lion at the tth iteration, respectively.

In this paper, b is equal to one that shows the logarithmic spiral shape and l is a random number that is obtained by $l = (a-1) \times rand + 1$, where a is determined by $a = -1 + t \times ((-1)/T)$. T indicates the maximum number of iterations. The order of using these paths by ants is determined by a Tabu list. A spiral path is selected in the first iteration and the ants are moved according to it. At the end of the iteration, the elite fitness and the name of the spiral path are stored in the memory of TS, the ants are a movement with a spiral path that is kept in memory until no progress is obtained for the fitness of *Elite*. The spiral path with the elite is placed in the Tabu list if the fitness of the Elite ant lion does not improve in two consecutive iterations, and then the next spiral path is chosen for the movement of ants. Therefore, the *Elite* can escape from the local optimal. This process is continued until the end condition is not met. The design and implementation of a Tabu list for ant's random walk are presented in Fig. 19.

Fig. 19 An example of a Tabu list

Spiral path				Elite fitness
AR	LO	RO	HY	Fit(*Elite*)

4.1.2 Ant lion relocations

ALO updates an ant lion based on a fitter ant. A fitter ant is an ant with better fitness compared to an ant lion, and so this ant is replaced with the ant lion in the next iterations. In this paper, all positions of ant lions are updated (i.e., relocation) based on their corresponding ants in *PAnt* and *Elite* ant lion. Based on the number of iterations, the fitnesses of ant and ant lion are updated to improve HH-ALO-Tabu's search ability. The proposed updating formula for the *i*th ant lion is explained as follows:

$$Al_i^{t+1} = Al_i^t + Y \times M_{internal}\left(Al_i^t - Al_{Elite}^t\right) + (1 - Y) \times M_{external}\left(Al_i^t - A_r^t\right), \tag{37}$$

$$Y = \frac{t}{T}, \tag{38}$$

$$\begin{bmatrix} M_{Internal} \\ M_{External} \end{bmatrix} = \begin{cases} \frac{F(Al_{Elite}^t)}{F(Al_{Elite}^t)+F(A_i^t)} & if\ Maximization, \\ \frac{F(A_i^t)}{F(Al_{Elite}^t)+F(A_i^t)} & if\ Maximization, \end{cases} \quad \begin{cases} 1 - \left(\frac{F(Al_{Elite}^t)}{F(Al_{Elite}^t)+F(A_i^t)}\right) & if\ Minimization, \\ 1 - \left(\frac{F(A_i^t)}{F(Al_{Elite}^t)+F(A_i^t)}\right) & if\ Minimization. \end{cases} \tag{39}$$

In the first iteration, the ant lions are relocated toward random ants from the population based on how they are strong (i.e., fitter or better) compared to ant lions (i.e., exploration). In the last iterations, the ant lions are relocated to the best ant lion (i.e., *Elite*) that is found so far (i.e., exploitation).

4.2 HH-ALO-Tabu

HH-ALO-Tabu is a hyper-heuristic composed of two stages. By using the DE algorithm, an optimal configuration is selected based on the CM, OBL, spiral path, and population ratio. Secondly, ALO-Tabu benefits from this optimal configuration by more efficiently solving the given problems.

In the first stage, the DE's population (*PDE*), ant's population (*PAnt*), and ant lion's population (*PAntlion*) are defined. Two maximum numbers of iterations [i.e., one for DE algorithm (*MaxiterDE*) and another for *PAnt* and *PAntlion* (*MaxiterAT*) populations] are defined. At the beginning of the algorithms, a set of parameters for both DE and ALO-Tabu including the size (i.e., *NS*, *NAnt*, and *NAntlion*), the dimension of *PDE*, *PAnt*, and *PAntlion* populations (i.e., dim_D, dim_{Ant}, and $dim_{DAntlion}$) are defined. For example, the population *PDE* with size *NS* and $dim_D = 4$ is generated using Eq. (40).

$$PDE_{ij} = LD_j + rand \times (UD_j - LD_j), \quad i = 1,\ 2, \ldots, NS, \quad j = 1,\ 2,\ 3,\ 4. \tag{40}$$

where LD_j and UD_j indicate the lower and upper boundary of the *j*th dimension, respectively. The chosen CM and the first spiral path are represented by the first and the second indexes PDE_i, respectively. The OBL approach and the ratio of the population are indicated by the third the fourth indexes, respectively. In this study, ten CMs, four spiral paths, and four OBL approaches are considered and so, $LD_j = 1$, $j = 1,\ 2,\ 3$ and $UD_1 = 10, UD_2 = 4$, and $UD_3 = 4$. In addition, the value of $PDE_{i4} \in [0,\ 1]$.

For example, $PDE_i = [4,\ 2,\ 3,\ 0.2]$ indicates that the fourth CM creates two populations *PAnt* and *PAntlion* with sizes *NAnt* and *NAntlion*. In addition, the third OBL method

is applied to consider 0.2% from *PAnt* and *PAntlion* and then compute the opposite populations (i.e., $\overline{PAnt}$ and $\overline{PAntlion}$). Finally, the second spiral path (i.e., LO) is used as a started update movement for ants. Now the fitness value for each ant in populations *PAnt* and $\overline{PAnt}$ should be obtained and the best ant (i.e., the current *PAnt*) from the union of *PAnt* and $\overline{PAnt}$ must be selected. The same process is done for *PAntlion* and $\overline{PAntlion}$ so that an appropriate *PAntlion* is obtained. Then, the best value of fitness *Fit*(*Elite*) and its corresponding solution *Elite* ant lion are found. Then, ants and ant lions are updated based on different spiral paths, TS (based on Sect. 4.1.1), and a new relocation equation (based on Sect. 4.1.2). respectively. The number of iterations (*MaxiterAT*) is checked as a stopping condition after all the solutions of *PAnt* and *PAntlion* are updated. Now, the best fitness value [i.e., *Fit*(*Elite*)] is considered as the fitness value for the current DE's solution [i.e., $Fit(PDE_k)$ where $k=1, 2,\ldots,NS$]. After all solutions in *PDE* are used, then the best DE's solution is determined. The final step in *PDE* is to update solutions based on DE operators such as crossovers, mutations, and selections until the number of iterations reaches the maximum.

A second stage starts with the PAnt and PAntlion populations corresponding to DE's best solution (i.e., the best configuration) as the initial populations. When the stop conditions for the population are met, the ALO-Tabu operators update the population until the stop conditions are met (in this case, the maximum number of iterations is set to *MaxiterAT*). The output is the best solution to the problem. Algorithm 4 illustrates the main steps involved in HH-ALO-Tabu's implementation.

Algorithm 4. HH-ALO-Tabu algorithm

Input: MaxiterDE, MaxiterAT, NS, NAnt, NAntlion, dim_D, dim_{Ant}, $dim_{Antlion}$, Cm, OBL, Sp *//Cm, OBL, and Sp are three lists that show the chaotic maps, OBL strategies, and spiral movement*
Output: PDE_best_solution
1 PDE= Initialize_DE (NS, MaxiterDE, dim_D) *//Initialize DE solutions or possible configuration based on Section 4.2*
2 For each solution in PDE do
3 {
4 [sel_cm, sel_OBL, sel_Sp]= Select_conf (Cm, OBL, Sp) *//Select chaotic maps, an OBL strategy, and a spiral movement from Cm, OBL, and Sp lists that are provided by the PDE solution*
5 [PAnt, PAntlion]= Generate_populations (NAnt, Antlion, dim_{Ant}, $dim_{Antlion}$, ratio, sel_cm, sel_OBL, sel_Sp) *//Generate population PAnt and PAntlion by the selected chaotic maps, perform OBL strategy to achieve appropriate ants and ant lions for initial population (based on Sections 2.4, and 2.5)*
6 Elite_Antlion= Find_Elite (PAnt, PAntlion) *//Find the the best solution*
7 i=1
8 While i<= MaxiterAT
9 {
10 A_i^{t+1} = Update_ant (sel_Sp, Elite_Antlion, A_{pbest}^{t}) *//Update i-th ant based Eq. (34)*
11 Al_i^{t+1} = Update_antlion (Elite_Antlion, A_i^t , i) *//Update i-th antlion based on Eqs. (37-39)*
12 Elite_Antlion= Find_Elite (PAnt, PAntlion) *//Find the best solution*
13 If ($i\%5==0$)
14 [S]= Check_improvement (Elite_Antlion) *//Check if Elite_Antlion is improved for two consecutive iteration*
15 If (S==false)
16 New_Sp =Add_Tabulist (Sp, Elite_Antlion, Sel_Sp) *//The selected spiral with Elite_Antlion is placed in Tabu list and a new spiral path is selected to perform the movement for ants*
17 Sel_Sp= New_Sp
18 i= i+1
19 } *//end while*
20 } *//end for*
21 Return PDE_best_solution

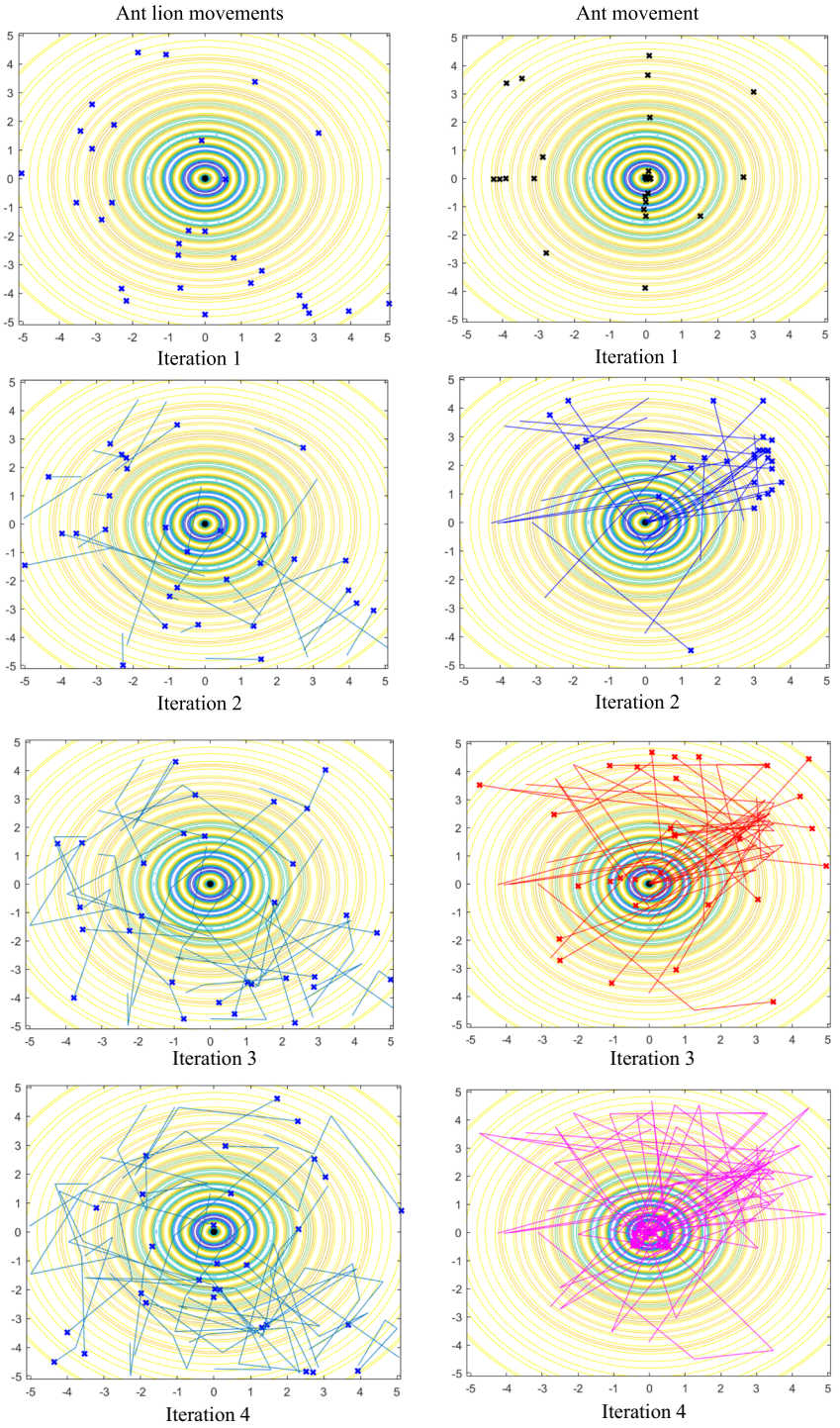

Fig. 20 The movements of HH-ALO-Tabu agents in drop-wave search space

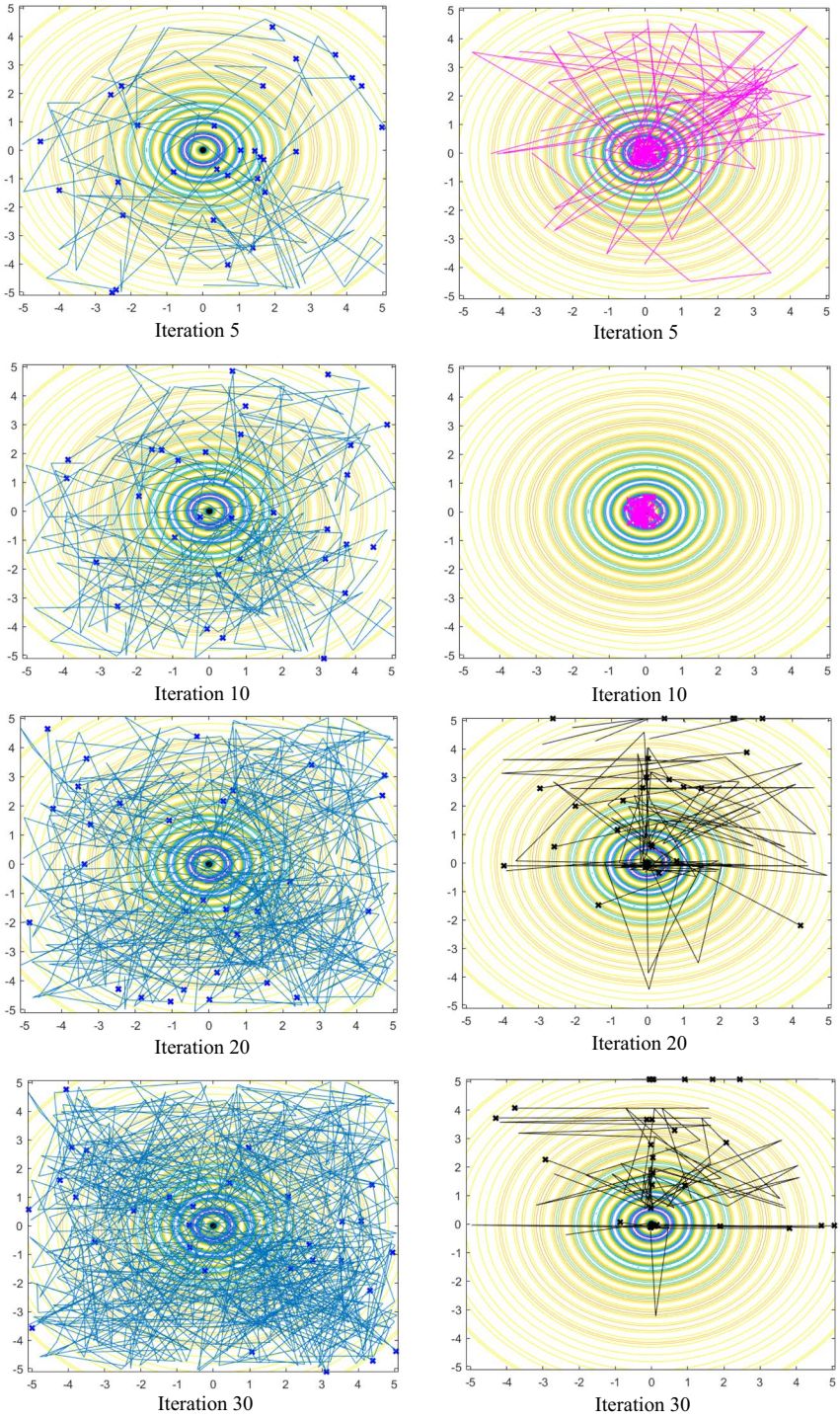

Fig. 20 (continued)

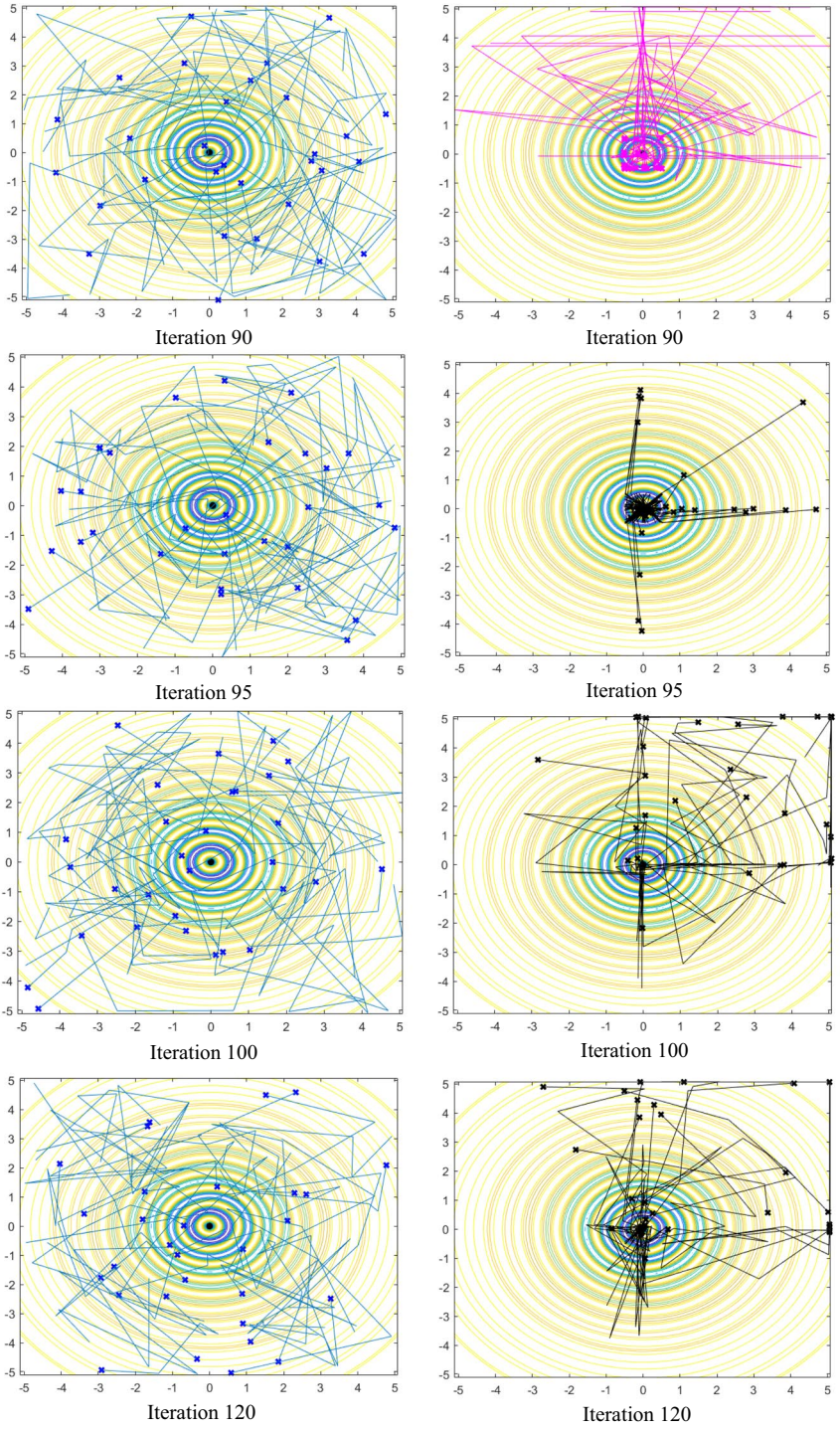

Fig. 20 (continued)

The movements of agents on drop wave function search space are examined to illustrate how HH-ALO-Tabu agents can converge toward an optimum solution. Figure 20 shows how agents from Ant and Ant lion populations are moved (i.e., update their locations) throughout the 120 iterations. In general, agents from the ant lion population tend to explore the search space while agents from the ant population tend to exploit the best solution (i.e., elite). It can be seen from Fig. 20 that agents of the ant population are moved with different colors (i.e., black, blue, magenta, and red) since they update their locations with different spiral paths based on this concept of whether a spiral can improve the elite or not. Guide for spirals and their colors are as follows: (1) Blue: Archimedes spiral, (2) Red: Hypotrochoid, (3) Black: Logarithmic spiral, and (4) Magenta: Rose spiral.

By iteration 10, the ants have found a near-optimal solution and have converged to a locally optimal solution. However, because the exploration of ant lions is not done this solution is not considered a final solution, and the searching is continued until iteration 150 where the optimal solution is determined.

In Fig. 21, HH-ALO-Tabu converges to global optimal while standard ALO finds a local optimal and traps there.

5 A new data replication algorithm: fuzzy hyper-heuristic ALO-Tabu (FHHAT)

Based on the HH-ALO-Tabu technique, we develop a multi-objective optimized replication algorithm (FHHAT). Its main objective is to provide the provider with both satisfactory performance and profitability. The data replication strategy is triggered when a job is entered at a particular site so the site can decide whether or not data replication is necessary. If replicating is necessary, how many replicas and where to store them are also the main issues dealt with in the data replication strategy to satisfy the SLA with the most amount of profit. In traditional distributed environments, most data replication algorithms generate so many replicas to enhance the utilization of the system. While

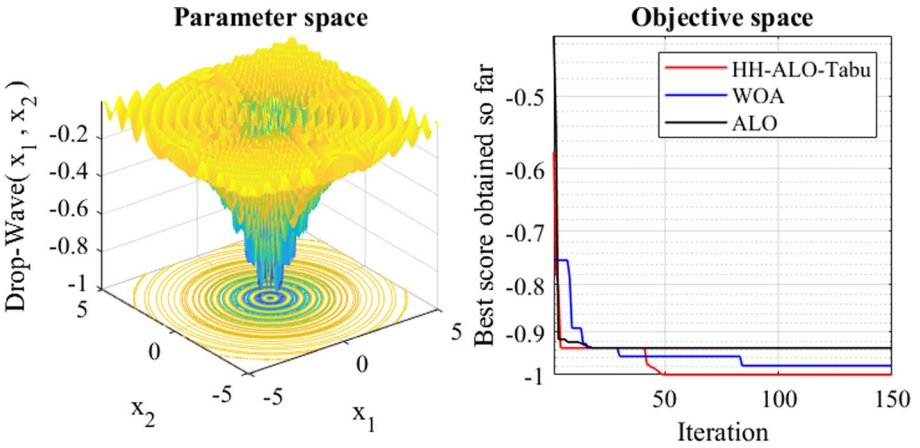

Fig. 21 Convergence curve of different methods for a drop-wave problem

storing unnecessary copies has an impact on the provider cost. Typically, the main goal of federated systems like data grids is to maximize utilization while minimizing loss in performance. A cloud environment may not be economically advantageous to the traditional replication algorithms, so they must be improved through improved optimization criteria.

5.1 Problem definition

Let us consider that there are a set of *Jobs* and a set of *Data Nodes* (*DN*) as shown in Fig. 22. Each DN is located in a region (in Fig. 22, there are three regions), and each job for execution needs some files. In Fig. 17, the broker assigns *Job2* to *DNm*+n in *Region2*. Data node *DNm*+n has *File2*, *File3,* and *File5*. But *File4* and *File8* don't exist in *DNm*+n, thus *Job2* must remotely access them, and then *File4* and *File8* should be replicated.

The proposed replication algorithm stores new replicas in such a way that the PEs, SLV, and mean service time (SMST) are minimized and SA is maximized.

Multiple objectives are involved in replica placement (PE, SLV, SMST, and SA), which may interact as well as conflict. There are also ambiguous dependencies between them that can be described using a fuzzy system. In this paper, the performance of the solution in terms of objectives (i.e., PE, SLV, SMST, and SA) is evaluated by a FIS. Thus, it is important to appropriate rules are selected for the FIS. In this article, the problem is divided into two parts: firstly, determine appropriate nodes for holding the replica files (Replica placement problem) and secondly determine appropriate rules for the FIS (Rule selection problem). Throughout the following sections, we will describe the details of the proposed algorithm, which we have called the fuzzy hyper-heuristic ALO-Tabu (FHHAT).

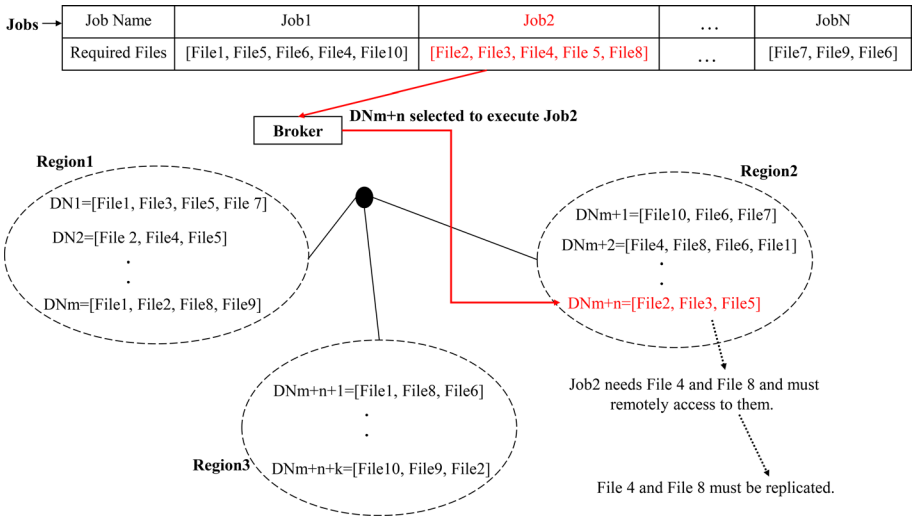

Fig. 22 Overview of replica placement problem

5.2 Objective functions

Many complex engineering fields have competing objectives, resulting in specific solutions that are detrimental to the other objectives, depending on the solution considered. As an example, there are functions that are maximized or minimized (e.g., maximizing profit and SA or minimizing resource consumption). In the replication process, it should be preferable to optimize the objective functions all at once but they compete with each other. We try to find an optimal solution according to the following significant parameters:

- *Service time* one of the main parameters in determining the throughput of a cloud that stores data is service time and delay in data access is used to evaluate the QoS. As a result, it is essential to take service time into account when replicating data.
- Availability of systems in a cloud environment is a key challenge since failures are more frequent than exceptional. So, the data replication algorithms must dynamically determine popular files and their location of them to satisfy data availability.
- *Load variance* overloaded conditions are common in large-scale systems but users expect rapid responses to requests that process large files. By distributing the load across the cloud system, data replication provides a solution for DN overload. The load variance represents the degree of load balancing the system during replication, so it is crucial to consider it.
- *Provider's expenditures* it is not economically feasible for the service provider to create a large number of replicas and hence economic cost management is needed. A dynamic data replication algorithm should strive to provide low response times along with provider benefits.

System availability (*SA*) the cloud aims to ensure high availability by storing copies of files on multiple DNs in a load-balanced manner. This is similar to grid technology (Mansouri 2014).

It is obvious that file f is considered unavailable only if all the replicas of f are not available (Wei Sun et al. 2012). Let variable $B(i, j)$ that has a value 1 when the file f_i $(1 \le i \le n)$ is stored on DN D_j $(1 \le j \le m)$, otherwise, it equals 0.

If FP_j indicates the failure probability for node D_j then the failure probability for node D_j. unavailable probability for file f_i can be obtained as Eq. (41):

$$PU(f_i) = \prod_{j=1}^{m\otimes} B(i,\ j) \times FP_j, \tag{41}$$

where $\prod^{\otimes}$ shows the cumulative multiplier of elements.

For example, let $i=1$, $m=4$, $FP_1=0.001$, $FP_2=0.005$, $FP_3=0.01$, $FP_4=0.002$, $B(1,\ 1)=1$, $B(1,\ 2)=0$, $B(1,\ 3)=0$, $B(1,\ 4)=1$ then $PU(f_1) = (0.001 \times 1) \times (0.002 \times 1) = 0.000002$.

The unavailability probability of file f_1 is 0.000002 and the availability of file f_i is found based on Eq. (42):

$$PA(f_i) = 1 - PU(f_i). \tag{42}$$

If all files of the system are available then the system is available. In the sequel, the SA can be calculated as follows and we intended to maximize this objective function.

$$\begin{aligned} SA &= \prod_{i=1}^{n} PA(f_i) \\ subject&: f_i > 0 \\ 0 &< PA(f_i) \leq 1. \end{aligned} \tag{43}$$

System mean service time (*SMST*) mean service time is one important aspect of QoS of the cloud and so we try to store necessary replicas in nodes with high performance. The proposed strategy pursues the objective of minimizing mean service time. The mean service time for file f_i on the node D_j can be computed based on Eq. (44).

$$STF = B(i, j) \times \frac{SF_i}{TR_j}, \tag{44}$$

where SF_i and TR_j indicate the size of file f_i and the transfer rate for node D_j, respectively. In addition, a Poisson process with a mean access rate $AR(i)$ is considered to model the requests to a file f_i. $AR(i)$ is determined based on Eq. (45).

$$AR(i) = \sum_{j=1}^{m} AN(i, j), \tag{45}$$

where $AN(i, j)$ shows the access rate for requests exporting from node D_j asking for file f_i. If file f_i is not available in the node D_j, then set $AN(i, j)$ equals zero. Hence, the mean service time of file f_i is determined based on Eq. (46).

$$\overline{STF}(i) = \sum_{j=1}^{m} \left(STF(i, j) \times \frac{AN(i, j)}{AR(i)} \right). \tag{46}$$

According to Long et al. (2014), the mean service time of the system is:

$$\begin{aligned} SMST &= \frac{1}{n} \times \sum_{i=1}^{n} \overline{STF}(i) \\ subject&: SF_i > 0, \\ 0 &< TR_j, \\ AN(i, j) &> 0, \\ AR(i) &> 0. \end{aligned} \tag{47}$$

System load variance (*SLV*) the third objective is load balancing which is the distribution of load among all the nodes. The load variance system is one of the commonly used metrics for evaluating the load balancing of a system, which is defined as the average of node loads. The proposed strategy pursues the objective of minimizing load variance. In (Phan et al. 2012) a load of f_i that is on the node D_j is obtained according to Eq. (48).

$$Load(i, j) = AN(i, j) \times STF(i, j). \tag{48}$$

Consider Eq. (45), the load of node D_j can be expressed by Eq. (49):

$$Load(j) = \sum_{i=1}^{n} Load(i, j). \tag{49}$$

We can obtain the average load of the system based on Eq. (50):

$$\overline{L} = \frac{1}{m} \times \sum_{j=1}^{m} Load(j), \tag{50}$$

Finally, we compute the load variance as follows:

$$\begin{aligned} SLV &= \sqrt{\frac{\sum_{j=1}^{m} (load(j) - \overline{L})^2}{m-1}} \\ &= \sqrt{\frac{\sum_{j=1}^{m} \left(\sum_{i=1}^{n} B(i,\ j) \times \frac{SF_i}{TR_j} \times AN(i,\ j) - \frac{1}{m} \times \sum_{j=1}^{m} \sum_{i=1}^{n} B(i,\ j) \times \frac{SF_i}{TR_j} \times AN(i,\ j) \right)^2}{m-1}} \\ &subject: \quad SF_i > 0, \\ &0 < TR_j, \\ &AN(i,\ j) > 0, \\ &AR(i) > 0, \\ &m > 1. \end{aligned} \tag{51}$$

Provider's expenditures (*PE*) although the main responsibility of the cloud provider is to provide services they want to achieve monetary profit since they deal with different costs such as employees' salary and power consumption. Some of these costs such as the cost of storage are directly affected by the data replication process and while some others such as onsite physical security may not be concerned by replication. Cloud providers have various expenditures for example they host several nodes to handle user requests. Power is consumed by these nodes, and other support hardware is required to make them work. Costs related to CPU, network bandwidth, and storage are taken into account when developing a data replication strategy.

CPU cost usually several nodes participate to execute the job. For example, the node that executes the job and the nodes that have the files needed to execute the job. Therefore, the CPU cost is proportional to the sum of the consumption of CPU time of all nodes (D_j) that are used in the execution and is obtained based on Eq. (52) (Sousa and Machado 2012). In general, a parameter is used as the unit cost of CPU resources to transform CPU time into the monetary cost.

$$\begin{aligned} &CPU\_\text{Cost} = \sum_{j} N_j \times \alpha_j \\ &subject: (1.25\ \$ \leq \alpha_j) \quad per\ 10^9\ MI, \end{aligned} \tag{52}$$

where N_j indicates the consumption of CPU time and α shows the unit cost of CPU that is dependent on the node and is expressed \$/million Instructions.

Network cost it is obvious that job execution in the cloud leads to some resource usage due to data transmission. Data transmission may be done between nodes that are placed in different locations of the cloud hierarchy. In addition, the transfer cost varies between nodes due to the heterogeneous structure of the cloud. For example, network cost between geographical regions is higher than the transfer cost inside a region. Costs associated with data transfer depend on bandwidth availability and pricing.

A job J needs a set of files $\{f_1, f_2,\ldots,f_j\}$ during execution. Some of these files may transfer from remote nodes $\{d_1, d_2,\ldots,d_k\}$ to the executor node D_i. So the network cost can be determined as follow (Ergezer et al. 2009).

$$Network\_Cost = \sum_{1}^{j}\sum_{1}^{k} f_j \times \beta_{n_i,n_k} \times d_{j,k}, \tag{53}$$

$$d_{j,k} = \begin{cases} 1 & if\ transferred\ from\ n_k\ to\ n_i, \\ 0 & Otherwise, \end{cases} \tag{54}$$

where β is the unit cost of bandwidth between any two specific nodes in the cloud and is expressed \$/GB.

Storage cost transmission of data uses bandwidth during the transfer, and storage is used at the destination. Therefore, transmission costs are related to bandwidth and storage costs. Each file may be used by several jobs during execution and a job that requires more data operations usually costs more. Using the parameter of the unit cost of storage, storage usage is converted to monetary cost. The storage cost can be obtained based on Eq. (55) (Tos et al. 2018).

$$\begin{aligned} &Storage\_Cost = \sum_{1}^{j}\sum_{1}^{k} f_j \times \delta_{n_k} \times \phi_{j,k} \\ &subject: (0.2\ \$\ \leq \delta_{n_k})\ per\ GB, \end{aligned} \tag{55}$$

$$\phi_{j,k} = \begin{cases} 1 & if\ f_j\ is\ read\ from\ node\ n_k, \\ 0 & Otherwise, \end{cases} \tag{56}$$

where δ shows the unit cost of storage for any specific cloud node and is expressed \$/GB. Finally, the estimated provider cost is obtained as follows and we intend to minimize it.

$$\begin{aligned} &Provider\_Expenditure = CPU\_Cost + Network\_Cost + Storage\_Cost \\ &subject: (1.25\$ \leq \alpha_j)\ per\ 10^9\ MI \\ &\qquad (0.2\$ \leq \delta_{n_k})\ per\ GB. \end{aligned} \tag{57}$$

5.3 Replica placement and rule selection steps

In this section, the proposed replication method (FHHAT) is explained, including the individual representation, objective functions, fitness evaluation, and updating of individuals (e.g., ants).

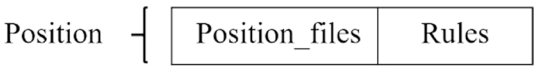

Fig. 23 Structure of an individual

5.3.1 Individual representation

For solving the defined problems (i.e., replica placement and rule selection) with the proposed HH-ALO-Tabu, the algorithm starts with two initial populations for *PAnt* and *PAntlion*. These two populations are determined with the best DE's solution (i.e., best configuration). Then, ants of *PAnt* and ant lions of *PAntlion* populations are updated using spiral movement and relocation strategy respectively until reaching the maximum number of iterations. The output shows the best solution for the data replication problem (i.e., the best place for replicas).

According to FHHAT, the population consists of individuals, and each individual is associated with a *Position* (which is divided into two distinct vectors, called *Position_files* and *Rules*). Figure 23 shows the structure of an individual. The number of copies and appropriate locations for storing them are represented in each *Position_files* of individuals. Using a FIS, the Rules vector will be used to calculate each individual's fitness. Each *Rules* vector length is equal to the number of valid rules that can be generated with PE, SA, SMST, and SLV. Each cell of the vector gets a value of 1 or 0. If a rule is selected, its corresponding cell in the vector is set to the value of 1 otherwise the cell value is set to 0.

With four parameters, (*SA*, *SLV*, *SMST*, and *PE*) 3^4 rules can be provided. In the presence of many rules, processing speed is slowed down, so only important rules can be used. For the selection of a good subset of rules from 81 possible rules, we define the *Rules* vector which is a binary vector $(x_1^t, x_2^t, \ldots, x_j^t, \ldots, x_K^t)$ where if rule j is to be included in the rule subset then $x_j^t = 1$, otherwise $x_j^t = 0$. Figure 24 shows the binary representation for the *Position* of one agent, which consists of the *Position_files* vector with four replicas for *File4* and $m+n+k$ DNs (where m DNs belong to *Region1*, n DNs belong to *Region2* and k DNs belong to *Region3*) and *Rules* vector. In the *Position_files* vector, the node name is in the first row and the location of replicas is represented in other rows. In the *Rules* vector, the first row indicates the number of rules.

In *Position_files*, the value of each bit (node) in a row can be 1 (i.e., it is a candidate to host a replica) or 0. Moreover, the number of ones in the ith row indicates the number of replicas for file *Filei*.

Position_files vector initialization in replica placement, we pursue a way to determine the best location for a new replica by evaluating each DN based on conflicting objectives such as minimization of response time and maximization of profit. As a result, placement problems can be viewed as an optimization problem. The *Position_files* of each individual shows a possible set of DNs for placing replica files. In this paper, we try to find a *Position_files* in the search space with the lowest fitness. We know that search space is nonlinear with many local minima.

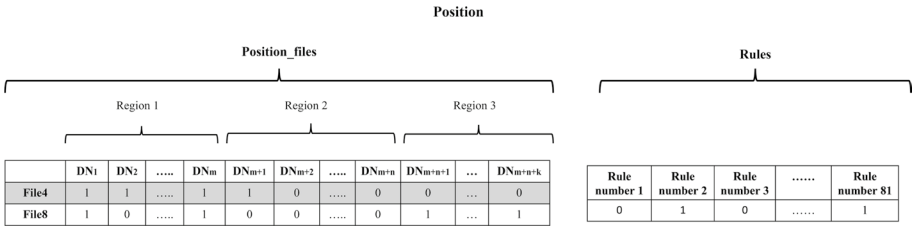

Fig. 24 Example for binary representation of *Position* for one agent

In this paper, we want to find appropriate DNs for placing replica files that minimize the cost where the *Position_files* vector of search individuals is positioned in a $m+n+k$-dimensional search space at positions in [0, 1]. By utilizing a limited search space (i.e., [0, 1]) and simpler operators, binary optimization algorithms offer better performance than continuous algorithms. In replica placement problems, the solutions (*Position_files* vectors) can be defined with the binary {0, 1} values and so a binary version of ALO is useful (Emary et al. 2016). At first, the Position-file vectors of ant and ant lion are initialized with continuous values, and then we squash them by the S-shaped function. The changing probability of elements from zero to one and vice versa is determined based on the squashing functions (i.e., transfer functions). Thus, the transfer function limits the movement of individuals to a binary space. One of the common transfer functions is the sigmoidal (S-shaped) function which is defined as follows (Emary et al. 2016).

$$x^t_{sig,j} = \frac{1}{1+e^{-10(x^t_{in,j})}}, \tag{58}$$

where $x^t_{in,j}$ and $x^t_{sig,j}$ indicate the continuous-valued input vector at jth dimension to sigmoidal function and the result from applying the sigmoidal function, respectively.

A threshold is required to reach the binary-valued output since the sigmoidal function is still in continuous mode. The most common stochastic threshold is used for ant [Eq. (59) and ant lion Eq. (60)] to obtain the binary solution. Figure 25 shows an example of the binarization procedure for *Position_files* of an individual (Emary et al. 2016).

$$A^t_{out,j}(Position\_files) = \begin{cases} 1 & if\ A^t_{sig,j}(Position\_files) \geq rand, \\ 0 & otherwise, \end{cases} \tag{59}$$

$$Al^t_{out,j}(Position\_file) = \begin{cases} 1 & if\ Al^t_{sig,j}(Position\_file) \geq rand, \\ 0 & otherwise, \end{cases} \tag{60}$$

where *rand* is a random number between 0 and 1. In Eq. (59), $A^t_{sig,j}(Position\_file)$ is a continuous value in the jth dimension of the *Position_file* vector from agents after applying the sigmoidal function and $A^t_{out,j}(Position\_file)$ is the binary version of this continuous value.

Rules vector initialization the *Rules* vector is a continuous vector with K length, in which K is the number of rules. *Rules* vectors need to convert to a binary version, the

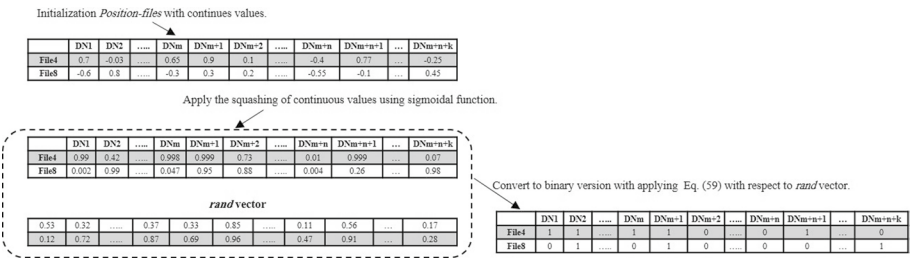

Fig. 25 An example of binarization for *Position_file* of one individual

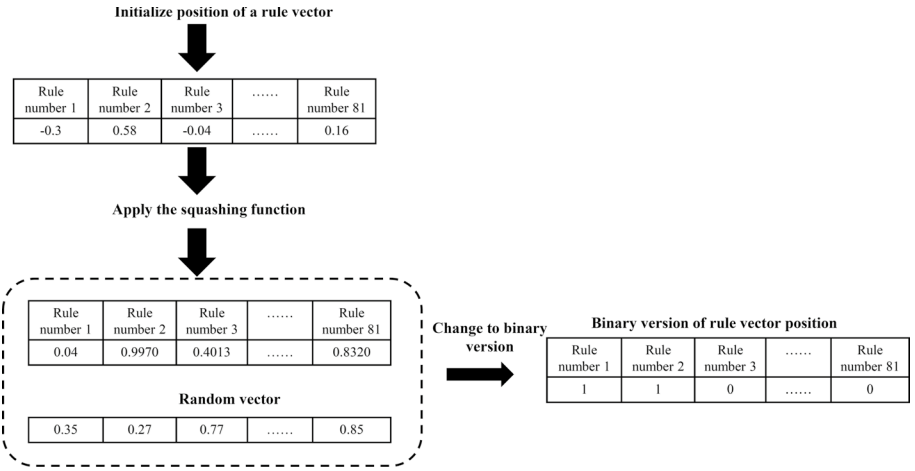

Fig. 26 An example of a *Rules* vector

Valid rules database				PE								
				L			M			H		
				SA								
				L	M	H	L	M	H	L	M	H
SLV	H	SMST	H	R1	R2	R3					R8	R9
			M	R10							R17	R18
			L	R19							R26	R27
	M		H	R28							R35	R36
			M									
			L									
	L		H									
			M								R71	R72
			L	R73	R74	R75					R80	R81

Fig. 27 An example of the database of valid rules

Table 5 An example of rules in the rule database

Rule	Objectives			
	SA	SMST	SLV	PE
R1	L (Low)	H (High)	H (High)	L (Low)
R2	M (Medium)	H (High)	H (High)	L (Low)
R3	H (High)	H (High)	H (High)	L (Low)
R80	M (Medium)	L (Low)	L (Low)	H (High)
R81	H (High)	L (Low)	L (Low)	H (High)

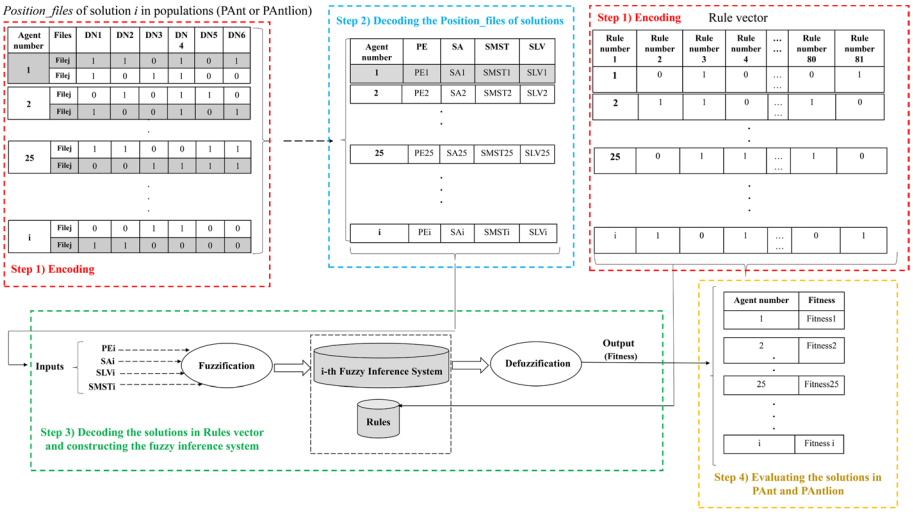

Fig. 28 General steps for rules determination

Agent number	Rule number 1	Rule number 2	Rule number 3	Rule number 4	… …	Rule number 80	Rule number 81
i	0	1	0	1	… …	0	1

Fig. 29 An example of the *Rules* vector of an agent

binary version for the *i*th rule vector can be obtained by Eq. (61). Value 1 means that a particular rule is selected, and 0 means that not selected. For example, in Fig. 26, *Rule 1* is selected but *Rule 2* is not selected.

$$A^{t}_{out,i,j}(Rules) = \begin{cases} 1 \;\; if \; A^{t}_{sig,i,j}(Rules) \geq rand, \\ 0 \;\; if \; A^{t}_{sig,i,j}(Rules) < rand. \end{cases} \tag{61}$$

Once the ant and ant lion positions have been defined, a multi-objective function must be used to calculate the fitness value of each agent. For most optimization studies, parameters were combined linearly, then weighted to determine which parameter was more significant for the user. In the replica placement problem, there are several conflicting goals for example response time and load variance must be minimized while some other objectives such as SA and profit of provider must be maximized. The decision-making process is complicated here, and it may even be impossible to find the optimal solution as quickly as possible that satisfies subjective preferences.

General phases for encoding, decoding, and utilizing the *Rules* vector are explained as follows.

1. Create a database of 81 rules. Figure 27 and Table 5 show an example of a rule database and some rules included in this database, respectively.

2. Encode the *Rules* vector of the solution as a binary vector and then decode it for the FIS. Figure 28 shows how a Rule vector is utilized to evaluate a solution in Position_files. This process consists of four main steps: (1) encoding, (2) decoding the *Position_files*' vector, (3) decoding the *Rules* vector and constructing a FIS, and (4) evaluating the solution in the *Position_file* vector.

Step 1 The population of ants and ant lions (i.e., PAnt and PAntlion) with size *i* are generated. Where the dimension of an individual in *Position_files* part is equal to the number of all DNs and in *Rules* vector is equal to the number of possible generated rules. In Fig. 28, the last agent has index *i*.

Step 2 *Position_files* of each solution are decoded and then objective functions (i.e., *SA*, *PE*, *SLV*, and *SMST*) for them are calculated.

Step 3 After calculating objective functions for each solution in *Position_files*, their fitness value of them must be computed (the details are in Sect. 5.3.2). Fitness for the *i*th solution is obtained with the *i*th FIS. Each FIS includes three main parts fuzzification, inference (i.e., fuzzy rules), and defuzzification. The rules for the *i*th FIS are obtained by the *i*th solution in the *Rules* vector. In Fig. 29 the rules for the *i*th FIS are rule number 2, rule number 4,…, and rule number 81 (cells that have value 1).

Step 4 Some rules in the inference system are triggered based on the values of each objective. The outputs of them are aggregated and then the output is returned as a number. The agent with the lowest value of fitness is considered an elite and the best solution.

The algorithm repeats these steps until it reaches its stop condition. The suggested DNs for placing replica files and selected rules by the best solution are considered the best placement and the most appropriate rules for the fuzzy system, respectively.

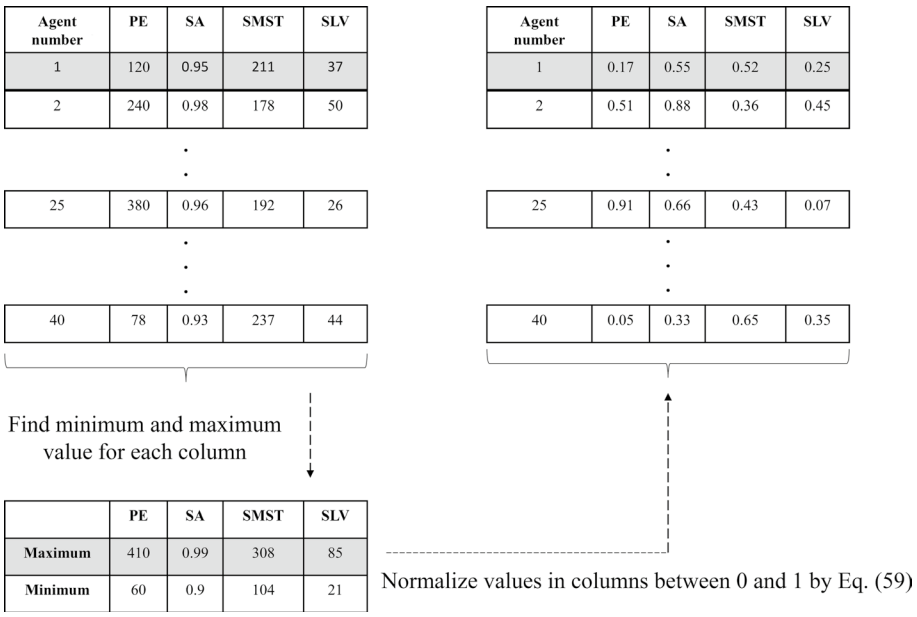

Fig. 30 Example for normalization objective functions

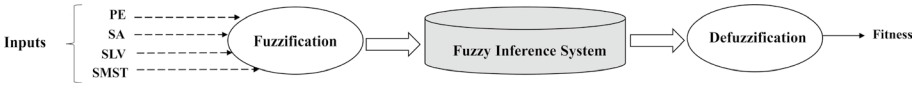

Fig. 31 Inputs and output for a fuzzy inference system

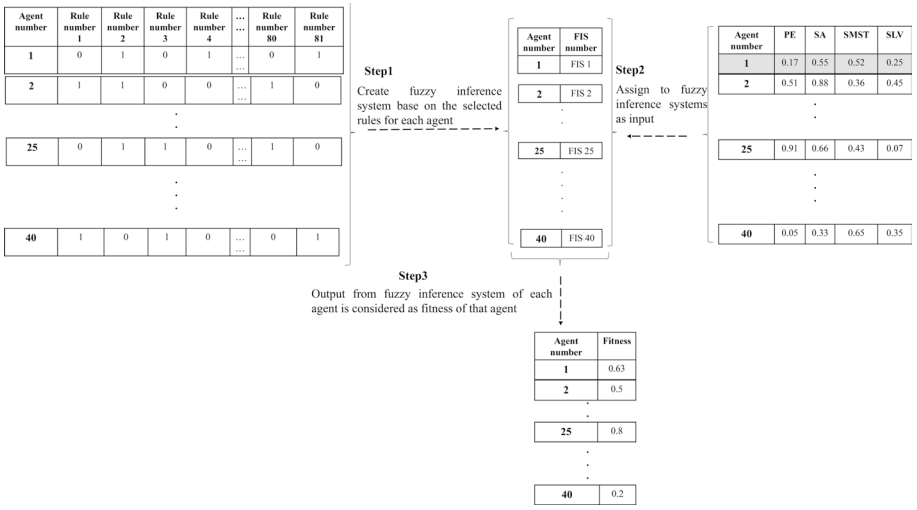

Fig. 32 A numerical example for the fourth step of fitness calculation

5.3.2 Fitness evaluation with fuzzy system

We show how the fitness of each solution (i.e., ant or ant lion) is calculated by utilizing the *Rules* vector in Sect. 5.3.1. Now we will explain steps 2 and 3 in more detail. In phase 2, the values of objective functions for each solution are calculated based on the *Position_files* of each agent.

In the second step, all objective function values for each solution are normalized between 0 and 1 based on the Min–Max normalization that is given by Eq. (62).

For mapping a v value from column A in range $[min_A, max_A]$ to a new range $[new\_minA, new\_maxA]$, the necessary computation is given as follows (Saranya and Manikandan 2013).

$$v_{normal} = \frac{v - \min_A}{\max_A - \min_A} \times (new\_\max_A - new\_\min_A) + new\_\min_A, \tag{62}$$

where v_{normal} indicates the new value in the particular range. Min–Max normalization has the advantage that all values are within a certain range. Figure 30 shows an example of normalization values for objective functions. In phase 3, for fitness calculation, we create a FIS based on selected rules from the *Rules* vector. For each solution, a FIS will be created by its *Rules* vector. After that, the values that are obtained from step two are used as input parameters for the created FIS.

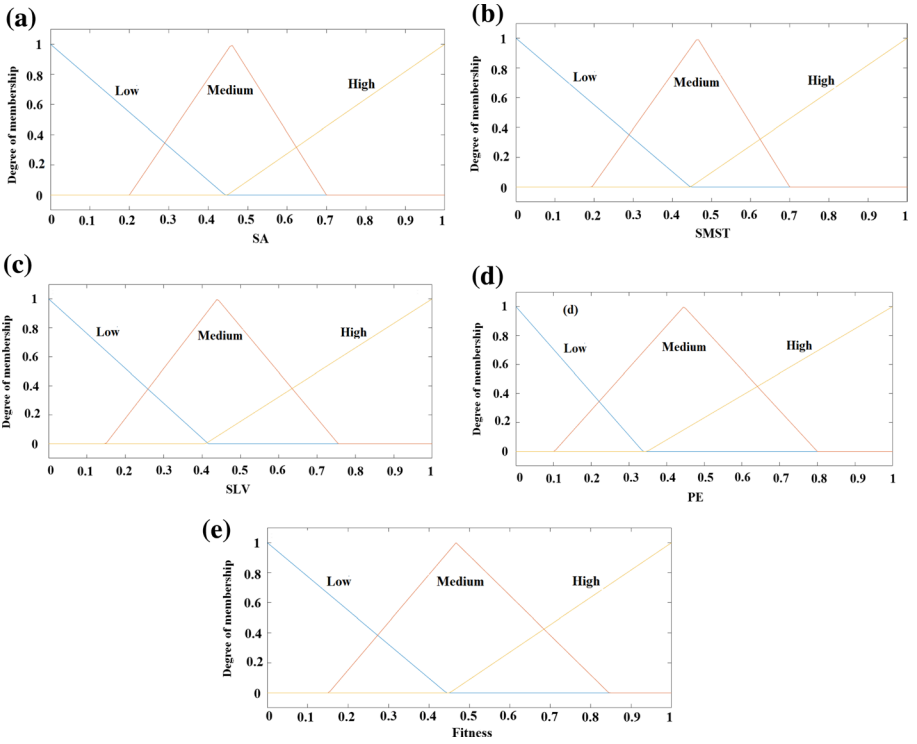

Fig. 33 Membership functions of *SA*, *SMST*, *SLA*, *PE*, and *Fitness*

Table 6 The fuzzy rules for evaluating fitness

SA	SMST	SLV	PE	Fitness
Low	Low	Low	Low	Low
High	Low	Low	Low	Low
High	Medium	Low	Low	Medium
Medium	High	High	Low	High
Medium	Low	Low	Low	Low
Medium	Medium	Low	Medium	Medium
Medium	Low	Low	Medium	High
Low	Medium	Medium	High	High
Low	High	High	High	High
High	Low	Medium	Medium	Medium
High	Medium	High	High	High
Medium	Low	Medium	Low	Medium
High	Low	Medium	Low	Low
Low	Low	Medium	Low	Low

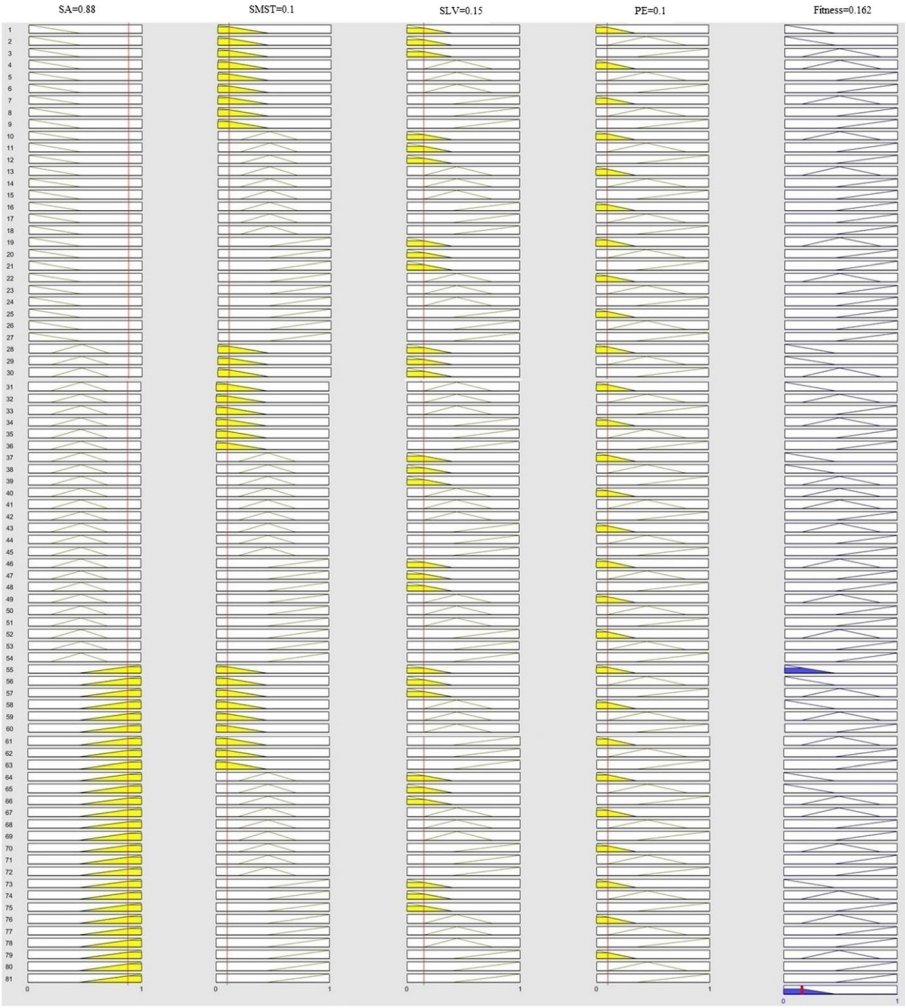

Fig. 34 An example of calculating fitness value with a fuzzy inference system

A FIS's output will be used to determine the fitness of a corresponding solution. Figure 31 shows the properties of a FIS including inputs and output. Figure 32 shows an example for step 3 of fitness calculation.

There are several fields in which FISs are suitable for making decisions, including robotics, pattern recognition, automatic control, decision analysis, etc. Fitness evolution uses fuzzy logic since it is conceptually simple, tolerates imprecise data, and can be built on expert experience. We use a common approach, namely Mamdani (Kacimi et al. 2020) to make a FIS since it can describe the expertise in more intuitive and widely applied in particular for decision support application. The most popular representation of knowledge is if/then rules. We consider triangle shapes to get the membership values during the fuzzification of inputs which is the main step in the creation of a fuzzy system. Based on the MATLAB-based fuzzy logic toolbox, Fig. 33 shows the three membership functions as *Low*, *Medium*, and *High*.

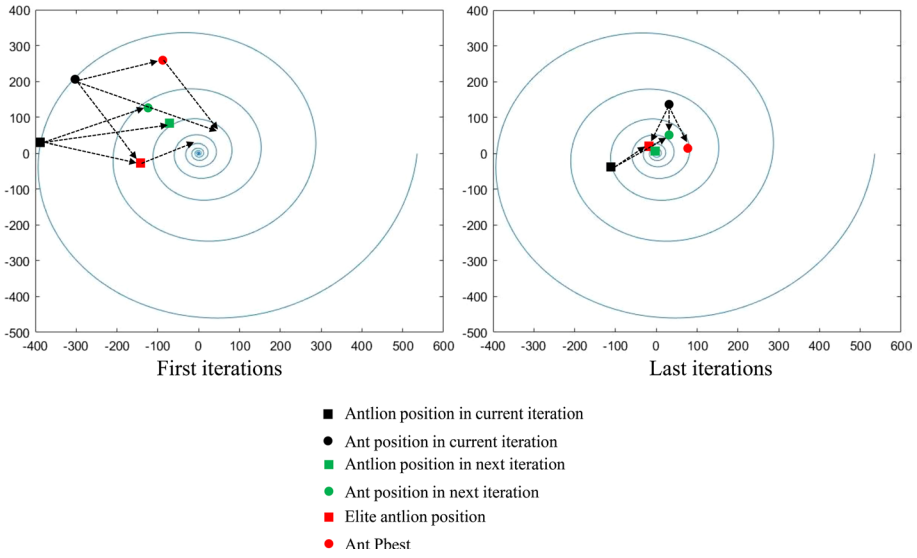

Fig. 35 An example of an ant lion and ant movement

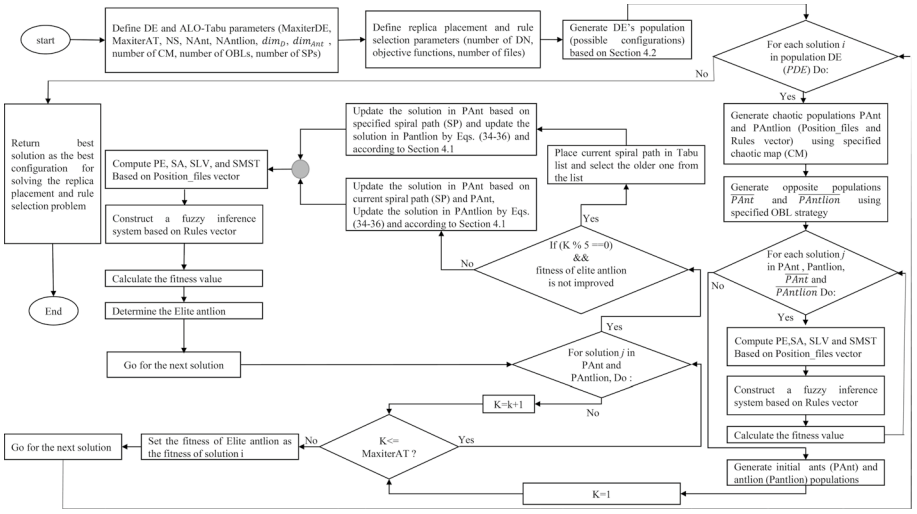

Fig. 36 Schema of proposed FHHAT

In Fig. 33, the SA of 0.6 shows a membership degree of 0.25 in the *High* interval, 0.4 in the *Medium* interval, and 0 in the *Low* interval. Figure 33e presents the output from zero to one. The proposed method selects 40 appropriate rules for our work and Table 6 indicates some of them. The agent with the lowest fitness in the *Rules* vector has the appropriate rule indexes.

Figure 34 shows an example for calculating the fitness by the defined FIS in a situation where SA, SMST, SLV, and PE are 0.88, 0.1, 0.15, and 0.1, respectively. It can be seen that the final value after aggregating the 81 fuzzy rules and using the defuzzification is equal to 0.162.

5.3.3 Updating the position of ants and ant lions

During the standard ALO, ants are moved towards an elite ant lion and a random ant lion. Thus, the next movement of the ant is highly dependent on the position of the random ant lion and if the selected ant lion is stuck on the local optimal then the ants move toward the local optimal. Unlike standard ALO, in the proposed replica method (FHHAT) ants move toward their best-experienced positions (which are updated at the end of each iteration) and elite ant lion. This mechanism helps ants move toward their best positions and universal best position. Thus, there is more chance for ants to reach the best solution. In the standard ALO, ant lion is updated only if a better (i.e., fitter) ant is found. In this situation, that ant becomes an ant lion and is replaced with that ant lion. One of the disadvantages of this strategy is reduced diversity in the population and as a result exploration ability is decreased. Instead of using the standard ALO, the proposed replica algorithm (FHHAT) relocates all ant lions based on corresponding ants and elite ants.

Figure 35 shows an example of how an ant lion and an ant might move along a logarithmic spiral path. The general steps of FHHAT for solving replica placement and rule selection problems can be seen in Fig. 36.

5.4 Replica replacement step

As replication has a limited number and storage size, it needs to be replaced to improve performance. If enough storage space exists in the best DN (BDN) that is selected by the FHHAT algorithm, the new replica is stored. Otherwise, some files must be removed based on two steps. The first step is to create a list of the least frequently used (LFU)

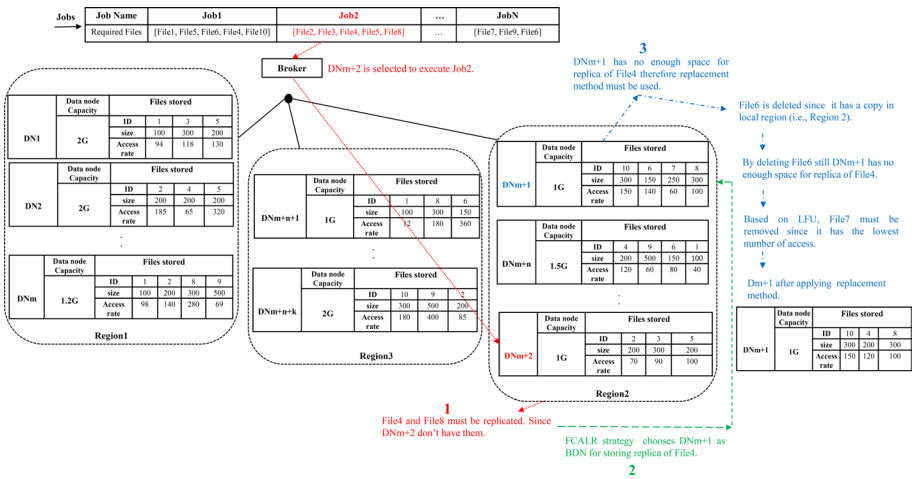

Fig. 37 An example of a replacement procedure

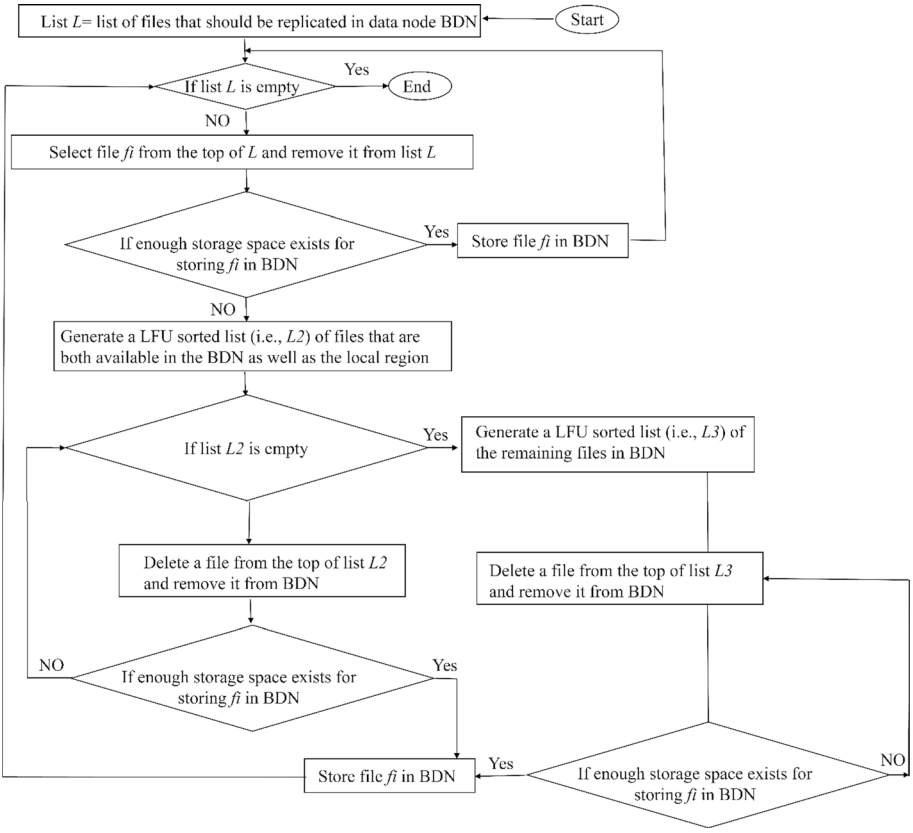

Fig. 38 Replica replacement flowchart

replicas available both in BDN and local regions. The next step is to delete files from the top of the list until enough space is available to create a new replica. In the second step, if space is still insufficient, the remaining files in the BDN are deleted based on the LFU strategy. In Fig. 37, *Job2* that needs *File2*, *File3*, *File4*, *File5*, and *File8* must be executed in $DNm+2$. So *File4* and *File8* should be replicated since they are not available in $DNm+2$. FCALR strategy determines $DNm+1$ as BDN for storing the replica of *File4*. But $DNm+1$ has not enough space for storing a replica of *File4* and so a replacement method must be triggered.

The replica replacement strategy tries to delete files that have at least a copy in the local region. In this example, *File6* is deleted since it has a copy in the local region (i.e., *Region2*) and we can access it with a high bandwidth link in the future. By deleting *File6* still, $DNm+1$ has not had enough space for a replica of *File4* and now with the LFU strategy, *File7* must be removed to provide sufficient storage.

Figure 38 shows different scenarios of the replacement process.

Table 7 Technique parameters and their values

Methods	Parameters	Values
SSA	G_c	1.9
	SF	18
	P_{dp}	0.01
	N_{fs}	4
WOA	P	0.5
	b	1
GWO	a	[2, 0]
MVO	Minimum WEP	0.2
	Maximum WEP	1
	Minimum TDR	0
	Maximum TDR	0.6
DE	Crossover probability	0.9
	Differential weight	0.5

Table 8 Unimodal functions (Mirjalili et al. 2014; Mirjalili and Lewis 2016)

Function	Dim	Range	f_{min}
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30, 200	[−100, 100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30, 200	[−10, 10]	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30, 200	[−100, 100]	0
$F_4(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2] [+(x_i - 1)^2]$	30, 200	[−30, 30]	0
$F_5(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	30, 200	[−100, 100]	0

Table 9 Multi-modal functions (Mirjalili et al. 2014; Mirjalili and Lewis 2016)

Function	Dim	Range	f_{min}
$F_6(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30, 200	[−600, 600]	0
$F_7(x) = \sum_{i=1}^{n} \left\lvert x_i^2 - 10\cos(2\pi x_i) + 10 \right\rvert$	30	[−5.12, 5.12	0
$F_8(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i-1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i-1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[−32, 32]	0

6 Numerical experiments and results

There are three parts to this section. In the first part, the experimental setup, benchmark functions, comparative algorithms, and evaluation parameters are described. As part of the second part, the proposed algorithm is evaluated on two benchmark functions and three engineering problems. A real-world cloud-based problem (i.e., data replication) is investigated in the last part of the paper.

Table 10 Composite functions (Mirjalili et al. 2014; Mirjalili and Lewis 2016; Jain et al. 2019)

Function	Dim	Range	f_{min}
$F_9(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	$[-65, 65]$	1
$F_{10}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	4	$[-5, 5]$	0.0003

6.1 Experimental setup

A 20-test function test suite taken from the standard CEC 2005 (Liang et al. 2005; Price et al. 2018) and CEC 2019 test suite (Suganthan et al. 2005) is used to evaluate the proposed algorithm. To terminate, the maximum number of iterations must be reached (i.e., *Maxiter* = 100). In addition, each technique is executed over 30 independent runs on MATLAB 2018 installed over Windows 10 (64 bits). Table 7 indicates the values of the controlling parameters for the compared techniques.

6.1.1 Benchmark functions

We evaluate HH-ALO-Tabu's performance using two types of benchmark functions (CEC 2005 and CEC 2019). As a result, CEC 2005 is characterized by three well-known types of test functions. These groups are as follows:

- *Unimodal* unimodal functions contain only one optimum (see Table 8), and are thus useful for evaluating how well a strategy is exploited and converged.
- *Multi-modal* there are multiple optimum values for these functions (see Table 9). One is the global optimum, and the others are local optimum values. A global optimum can only be obtained by escaping all local minima. So multi-modal functions are used to evaluate the exploration of algorithms.
- *Composite* a combined, rotated, shifted, and biased version of the two previous categories (see Table 10) is shown in Table 10. In these simulations, there are a great

Table 11 The 100-digit challenge basic test function (Suganthan et al. 2005)

No	Functions	$F_i^* = F_i(x^*)$	Dim	Search range
cec1	Storn's Chebyshev polynomial fitting problem	1	9	$[-8192, 8192]$
cec2	Inverse Hilbert matrix problem	1	16	$[-16384, 16384]$
cec3	Lennard–Jones minimum energy cluster	1	18	$[-4, 4]$
cec4	Rastigin's function	1	10	$[-100, 100]$
cec5	Griewangk's function	1	10	$[-100, 100]$
cec6	Weierstrass function	1	10	$[-100, 100]$
cec7	Modified Schwefel's function	1	10	$[-100, 100]$
cec8	Expanded Schaffer's F6 function	1	10	$[-100, 100]$
cec9	Happy Cat function	1	10	$[-100, 100]$
cec10	Ackley function	1	10	$[-100, 100]$

number of local minima, which is a real problem. These functions can be better balanced with algorithms by establishing a better balance between exploration and exploitation.

In addition, 10 modern CEC 2019 benchmark functions in Table 11 (i.e., The 100-Digit Challenge) are applied as an extra evaluation on HH-ALO-Tabu.

6.1.2 Comparative algorithms and performance measures

Two types of methods are used to evaluate the proposed HH-ALO-Tabu algorithm:
(1) four state-of-the-art algorithms:

- *WOA* (Mirjalili and Lewis 2016) this algorithm is derived from whale behavior when hunting.
- *GWO* (Mirjalili et al. 2014) this algorithm is based on gray wolves' foraging behavior.
- *Squirrel search algorithm*, *SSA* (Jain et al. 2019) a model describing southern flying squirrel locomotion and foraging behavior.
- *Multi-verse optimizer*, *MVO* (Mirjalili et al. 2016) this algorithm uses the White hole, the Black hole, and the Wormhole in cosmology.

(2) four variants of the ALO algorithm:

- *ALO* (Mirjalili 2015) as discussed in Sect. 2.3, it emulates the behavior of the ant lion during hunting.
- *GA-ALO* (Emary et al. 2016) it is a standard ALO with a crossover and mutation operator.
- *Chaotic-ALO* (Zawbaa et al. 2016) it uses a tent function to provide diversity in the population.
- *OBL-ALO* (Wang et al. 2020) it considers the OBL schema in the ALO algorithm to escape from local stagnation.

To assess the HH-ALO-Tabu's performance against the comparative algorithms, two types of measures are used: qualitative and quantitative. Qualitative includes five metrics as follows:

- *Search history* the ant lions position throughout all iterations.
- *Convergence* the elite fitness throughout all iterations.
- *The trajectory of the first agent in its first dimension* it shows the exploring search space ability.
- *The average fitness of all agents* it presents stability and convergence ability.
- *Exploration vs. Exploitation percentage* it represents the difference between dimensions of agents.

In this paper, a dimension-wise diversity measurement [Eq. (63)] is used to show the diversity of the swarm in each iteration. Then, *Div* is obtained by the average diversity of all dimensions.

$$Div_j = \frac{1}{n}\sum_{i=1}^{n} median(x^j) - x_i^j, \tag{63}$$

$$Div = \frac{1}{D}\sum_{j=1}^{D} Div_j, \tag{64}$$

where median (x^j) and x_i^j indicate the median of dimension j in the swarm and dimension j of agent i, respectively. The swarm size is represented by n.

The percentage of exploration and exploitation in each iteration is determined as follows:

$$Xpl\% = \frac{Div}{Div_{\max}} \times 100, \tag{65}$$

$$Xpt\% = \frac{\left|Div - Div_{\max}\right|}{Div_{\max}} \times 100, \tag{66}$$

where Div and $Div_{\max}$ show the diversity of swarm and the maximum diversity in all iterations, respectively. For an iteration, the exploration and exploitation percentages are presented by $Xpl\%$ and $Xpt\%$ are exploration and exploitation percentages for an iteration, respectively.

Our goal is to understand how the algorithm explores and exploits the search space by tracking the position of the ant lion. In addition, the level improvement of the found global optimum is measured by elite fitness during optimization. In the quantitative measure, the experimental results will be described based on the average (AVG), standard deviation (STD), and Wilcoxon signed-rank test statistic of the function values.

(a) *Average Average*(x) indicates the average of obtained outcomes as follows:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{67}$$

(b) *STD* it quantifies the variation of a set of data and can be computed based on Eq. (68).

$$STD = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}. \tag{68}$$

(c) *Wilcoxon signed-rank test* it describes the difference between the two samples and answers the following hypotheses:

$$\begin{aligned} H_0 &: mean(A) = Mean(B), \\ H_1 &: mean(A) \neq mean(B), \end{aligned} \tag{69}$$

where A and B show the results of the first and second methods, respectively.

In addition, it can investigate whether one method outperforms the other. In Eq. (70), d_i indicates the difference between the performance scores of two methods for solving the ith out of n problems. R^+ shows the sum of ranks for the problems in such a way the first method outperforms the second and R^- denotes the sum of ranks in such a way the second

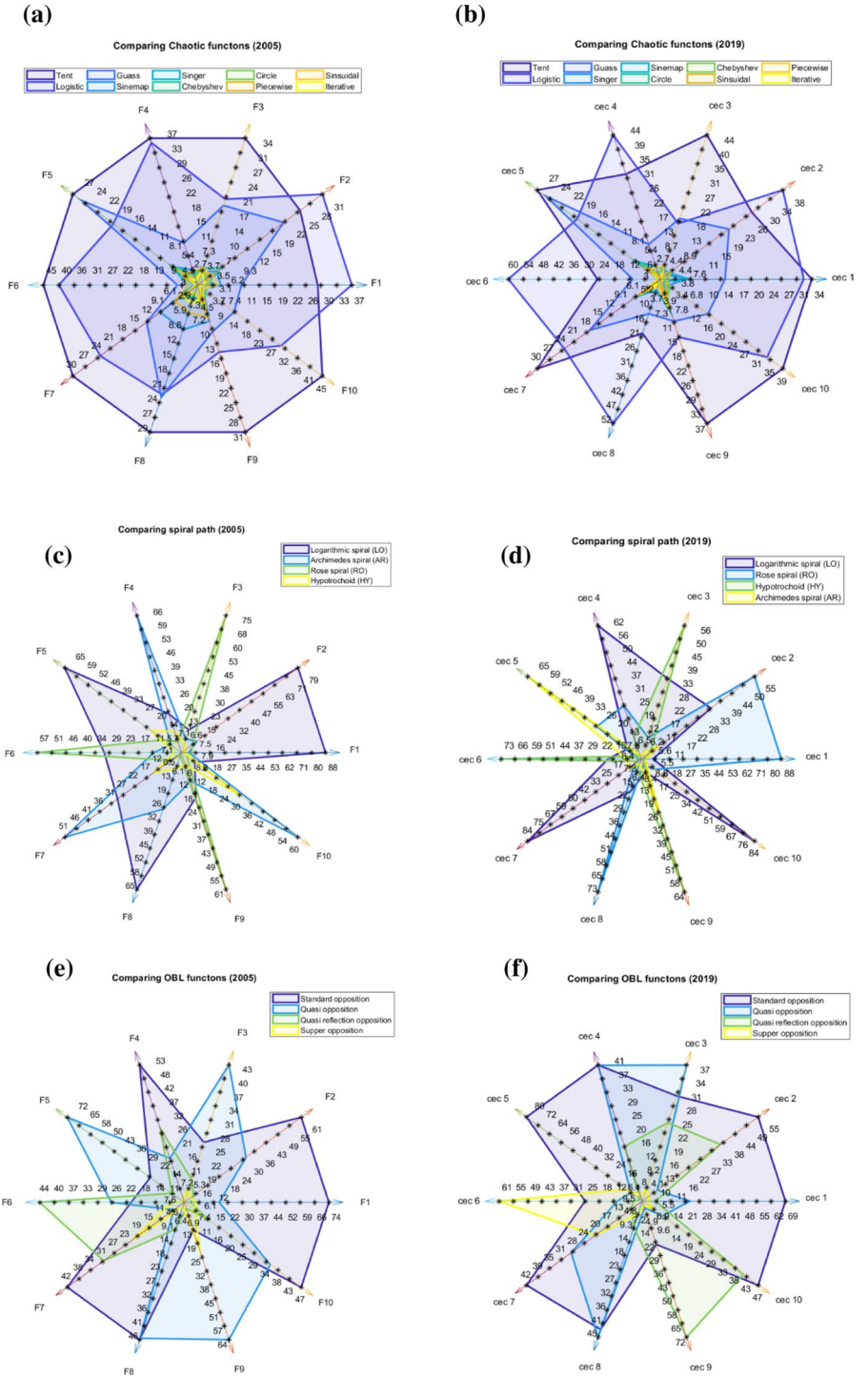

Fig. 39 The probability of occurrence for different chaotic maps, OBL strategies, and spiral paths

method outperforms the first. One of them is disregarded when these sums are an odd number.

$$R^{+} = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i), \tag{70}$$

$$R^{-} = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i). \tag{71}$$

We try to determine the p-value for comparing the method based on a significant level $\alpha = 0.05$ and if the p-value is lower than α, then the null hypothesis is rejected.R^{+} indicates a high mean value and superiority over other methods for various groups of experiments.

6.2 Experimental series

First, the best configuration has been determined. Then, we study the performance of proposed HH-ALO-Tabu in solving benchmark functions with two experimental series, first based on CEC 2005 and with ALO variants, and second based on CEC 2019 and with a recently introduced meta-heuristic.

6.2.1 Analysis of best configuration

We present the average occurrence of each component among the best solutions to illustrate how each component affects the performance of the ALO-Tabu method. According to Fig. 39, the spiral path and CM are most likely to occur in the best solution. In addition, Fig. 39a and b respectively indicate the average probability of occurrence for each CM along with CEC 2005 and CEC 2019 benchmark functions. There is no doubt that the Tent map has the greatest occurrence of all 20 test functions, followed by the logistic and Gauss maps. In addition, the Tent map achieves the best solution in 32.8 iterations of CEC 2005 functions and 55.5 iterations of CEC 2019 functions. This parameter for logistic and Gauss map for CEC 2005 functions is 31.25 and 25.75, respectively. We can see that the standard OBL technique obtains the highest occurrence probability for six functions (i.e., $F1$, $F2$, $F4$, $F7$, $F8$, and $F10$) in CEC 2005 and five functions (i.e., $cec1$, $cec2$, $cec5$, $cec7$, and $cec10$) in CEC 2019. In addition, the standard OBL technique can obtain its high frequency at $F8$ and $F10$ which are 90 and 100, respectively.

The QOBL is in the second place of occurrence for five benchmarks (i.e., $F3$, $F5$, $F9$, $cec3$, and $cec8$), followed by QROBL with three benchmarks (i.e., $F6$, $cec4$, and $cec9$) and the Super OBL method with one function (i.e., $cec6$). For the HH-ALO-Tabu algorithm, the standard OBL and QOBL have been preferred since they show a high probability of occurrence.

The average probability of occurrence for each spiral path along for CEC 2005 and CEC 2019 benchmark functions can be seen in Fig. 39e and f, respectively. It can be seen that all of the spiral movements are used in exploring the search space for finding the best solution in two types of functions but the Logarithmic and Hypotrochoid are employed in more iterations compared to two other spirals for finding the best solutions (in average 7.5%, 4.5% for CEC 2005 and 5.25%, 6.2% for CEC 2019). This result shows the effectiveness of using the Logarithmic, Hypotrochoid, and Archimedes spiral for CEC 2005 is more than

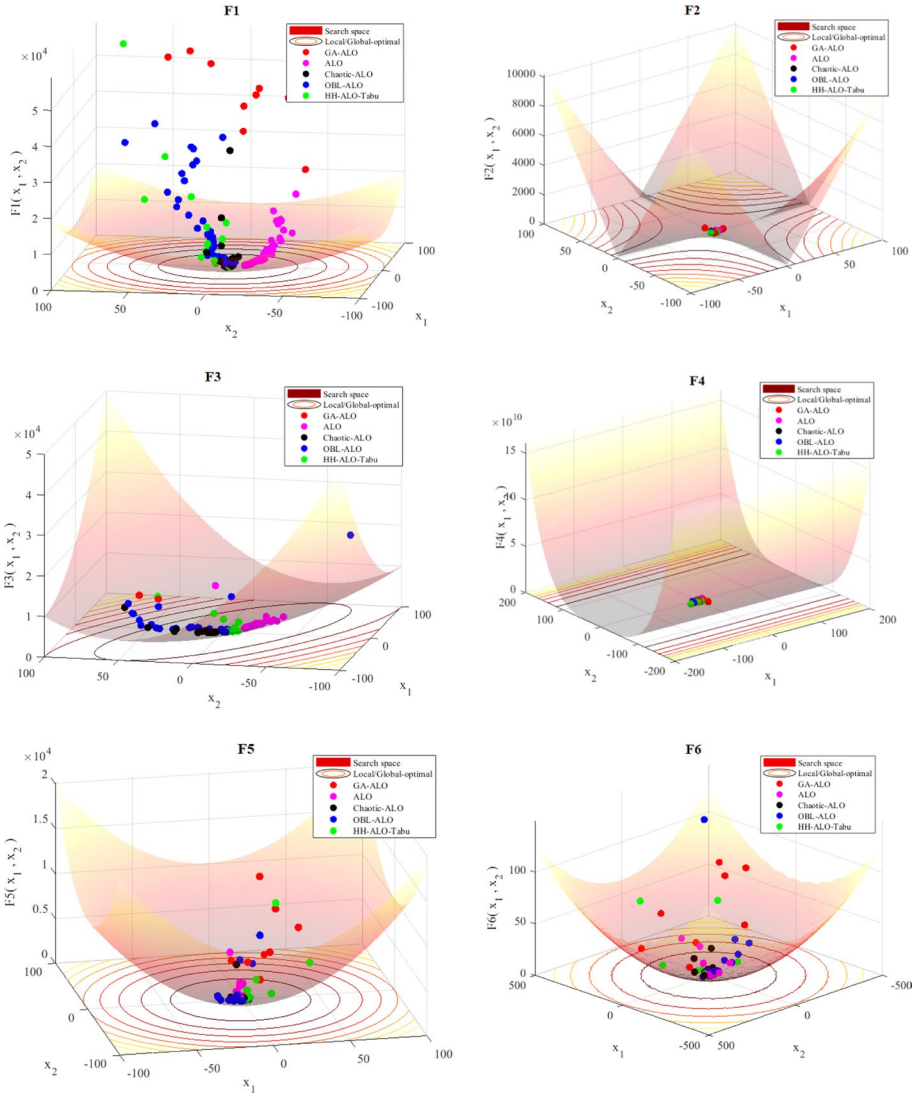

Fig. 40 Search history of agents during solving the CEC 2005 functions

the Rose spiral. While for CEC 2019 Rose spiral along with Logarithmic and Hypotrochoid are more effective than Archimedes spiral.

6.2.2 Comparison with ALO variants

To determine how search agents move through space, converge towards promising areas, and improve fitness values during the optimization process, a qualitative study is conducted. Therefore, the test functions with two dimensions and 100 search agents are considered to study the behavior of agents. As shown in Fig. 40, each search agent has a

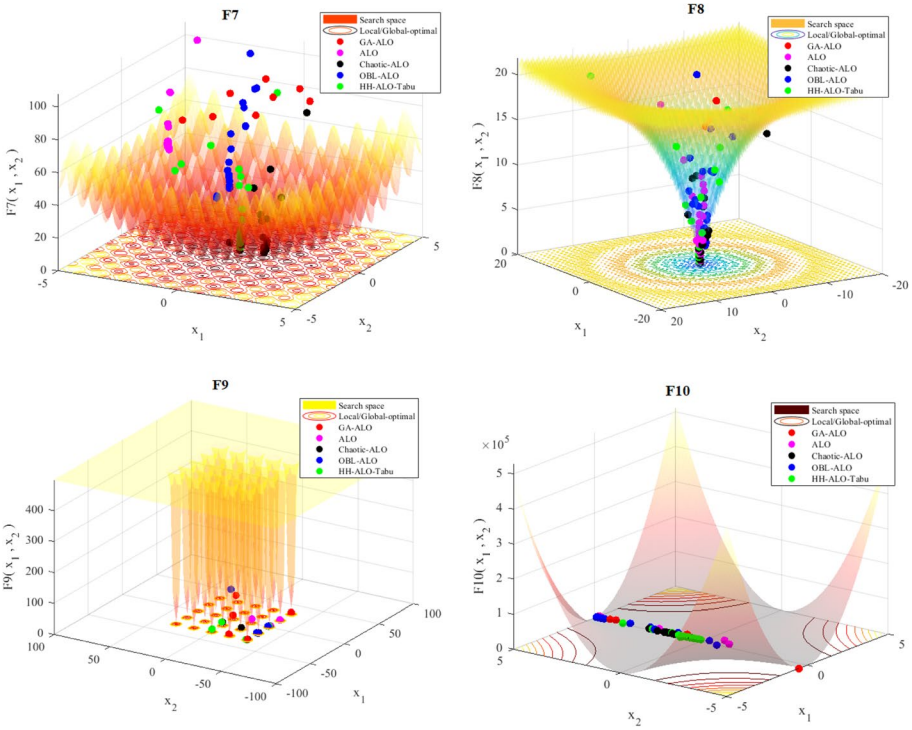

Fig. 40 (continued)

history of searches. Based on the number of sampled points near the global optima, the HH-ALO-Tabu strategy searches around the promising areas of the search space.

From Fig. 40, it can be seen that GA-ALO and ALO failed to find the global optimal in some functions (i.e., *F*4, *F*6, and *F*1) and it shows that the agents failed to search entire the search space properly and they are trapped into local optimal. Although evolutionary operators (crossovers and mutations) improve ALO's search capability, they are not enough to exploit the most promising regions.

The fluctuations of the first agents from different methods in the first dimension are represented in Fig. 41. Depending on how agents move, methods can be divided into two categories: (1) methods that have abrupt changes (i.e., Chaotic-ALO, HH-ALO-Tabu, and OBL-ALO). (2) Methods that have no significant abrupt changes throughout the iterations. Based on Fig. 41, it can be seen that most search agents (including the proposed method, Chaotic-ALO, and OBL-ALO) experience abrupt fluctuations during the early stages of optimization. The agents explore the search area and then move toward the most promising regions after they have discovered the best region. In *F*4, *F*7, and *F*9, the other methods (i.e., OBL-ALO and Chaotic-ALO) which focus on generating good initial solutions cannot provide a balance between exploration and exploitation.

The OBL strategies are generally used to generate an initial population near to global optimal as much as possible and hence it has a better performance compared to Chaotic-ALO in unimodal functions (i.e., *F*1, *F*2, and *F*4). On the other hand, Chaotic-ALO has

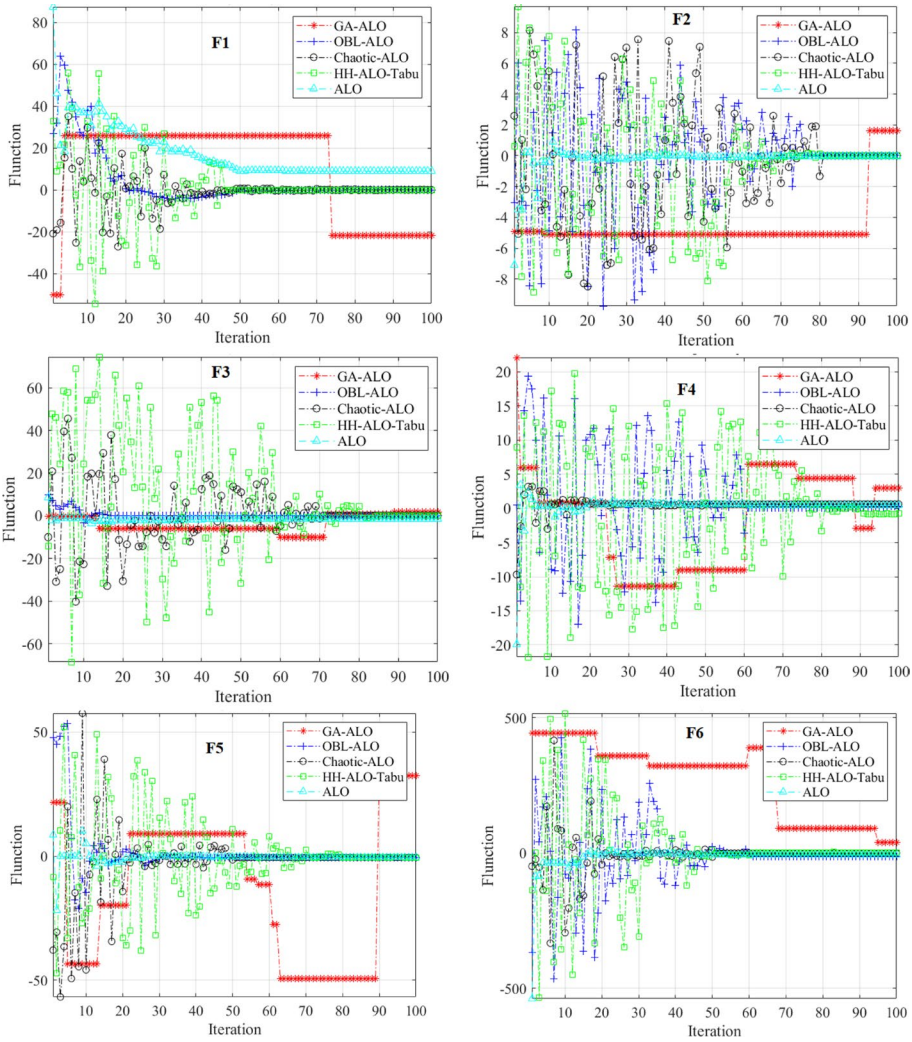

Fig. 41 Trajectory of the first variable in the first search agent (for CEC 2005 functions)

better performance in multimodal functions (i.e., $F8$, $F9$, and $F10$) compared to OBL-ALO since it has a strong exploration mechanism.

To enhance the diversity of solutions, it is necessary to provide suitable initial solutions for an optimization algorithm as well as a strong exploration mechanism. The movement style of the HH-ALO-Tabu's agent shows that the proposed algorithm performs more exploration in complex functions and does more exploitation in unimodal functions.

For different methods, the trajectory curve (Fig. 41) shows how agents move over time. Then, Fig. 42 indicates the exploration and exploitation plot to show the behavior and movement style of the whole swarm. It can be seen from Fig. 42 that the average exploration and exploitation percentage of HH-ALO-Tabu for $F8$, $F9$, and $F10$ until the

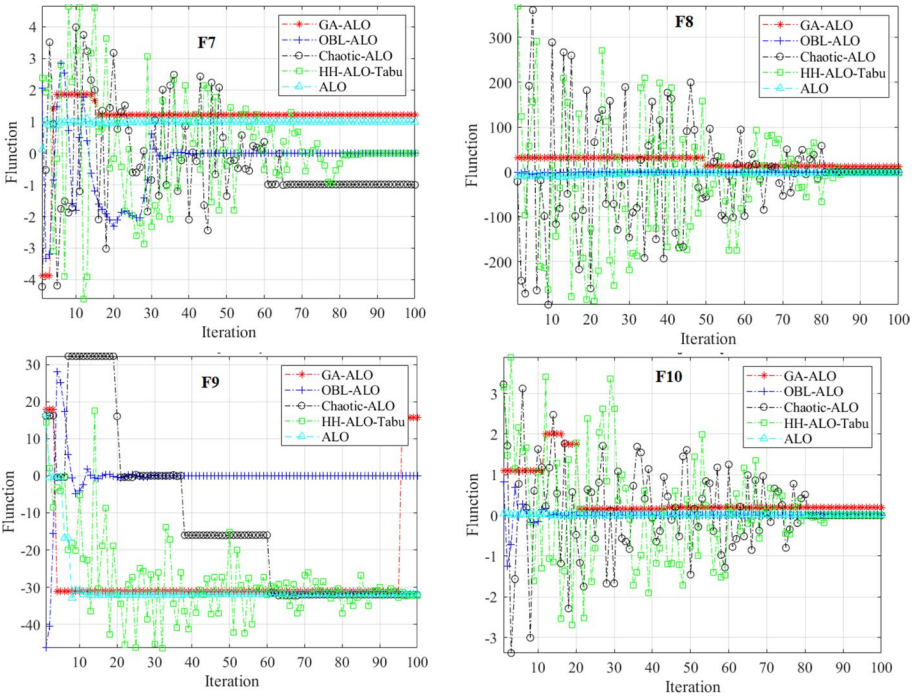

Fig. 41 (continued)

60th iteration is equal to [52–48%], [60–40%], and [55–45%], respectively. We know that the complex functions have several local minimal and hence the algorithm must explore the search space more than exploiting the best solution. This result shows that agents of HH-ALO-Tabu are far apart from each other until the 60th iteration approximately and after that the distance between them is decreased. Considering $F2$ and $F4$ in Fig. 42, it can be interpreted that OBL-ALO performs exploration less than 50% of the time and the rest of the time it performs the exploitation, While Fig. 41 shows that the first agent changes its location most of the time and its location doesn't change less than 50% of the time. As a result, it is important to investigate the behavior of agents individually and in swarms.

Figure 43 illustrates how the best solution improved during optimization. We can see the descending trend of all methods. However, the HH-ALO-Tabu algorithm's capability to approximate the global optimal solution for the test functions is more evident. It can be seen that in unimodal functions in which we have one global optimal the convergence curve of most methods is soft (i.e., without steps part). In complex functions (i.e., $F6$–$F10$), those methods with stepwise behavior (i.e., chaotic-ALO and HH-ALO-Tabu) achieve better results and they explore the search space better than ALO and OBL-ALO. The GA-ALO algorithm has a good explore capability but it can be seen that it fails to reach an optimal solution for $F5$, $F7$, and $F8$. A good initial population seems to be necessary for the first iteration for the algorithm to reach a better outcome.

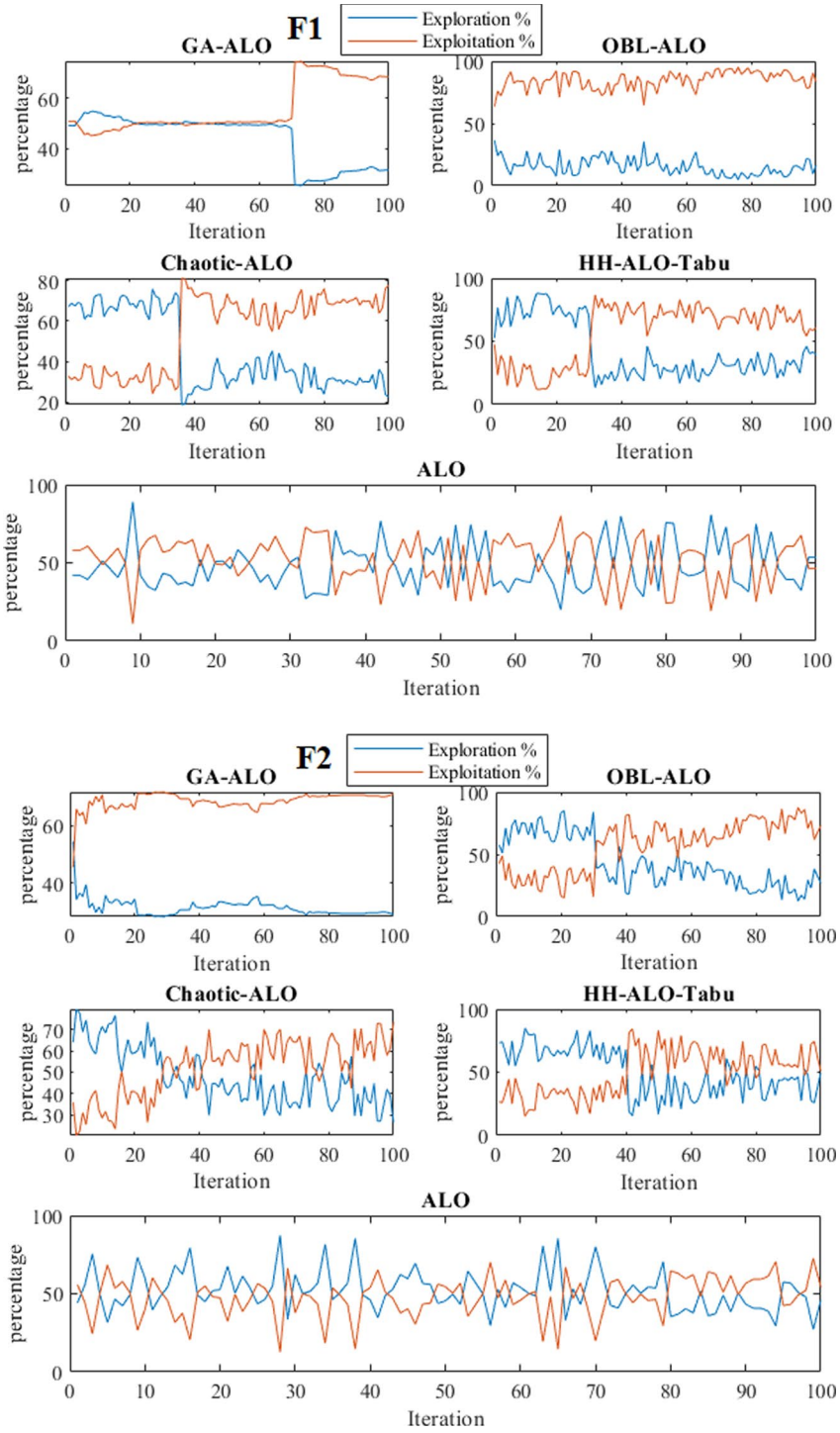

Fig. 42 Exploration vs. exploitation rate (for CEC 2005 functions)

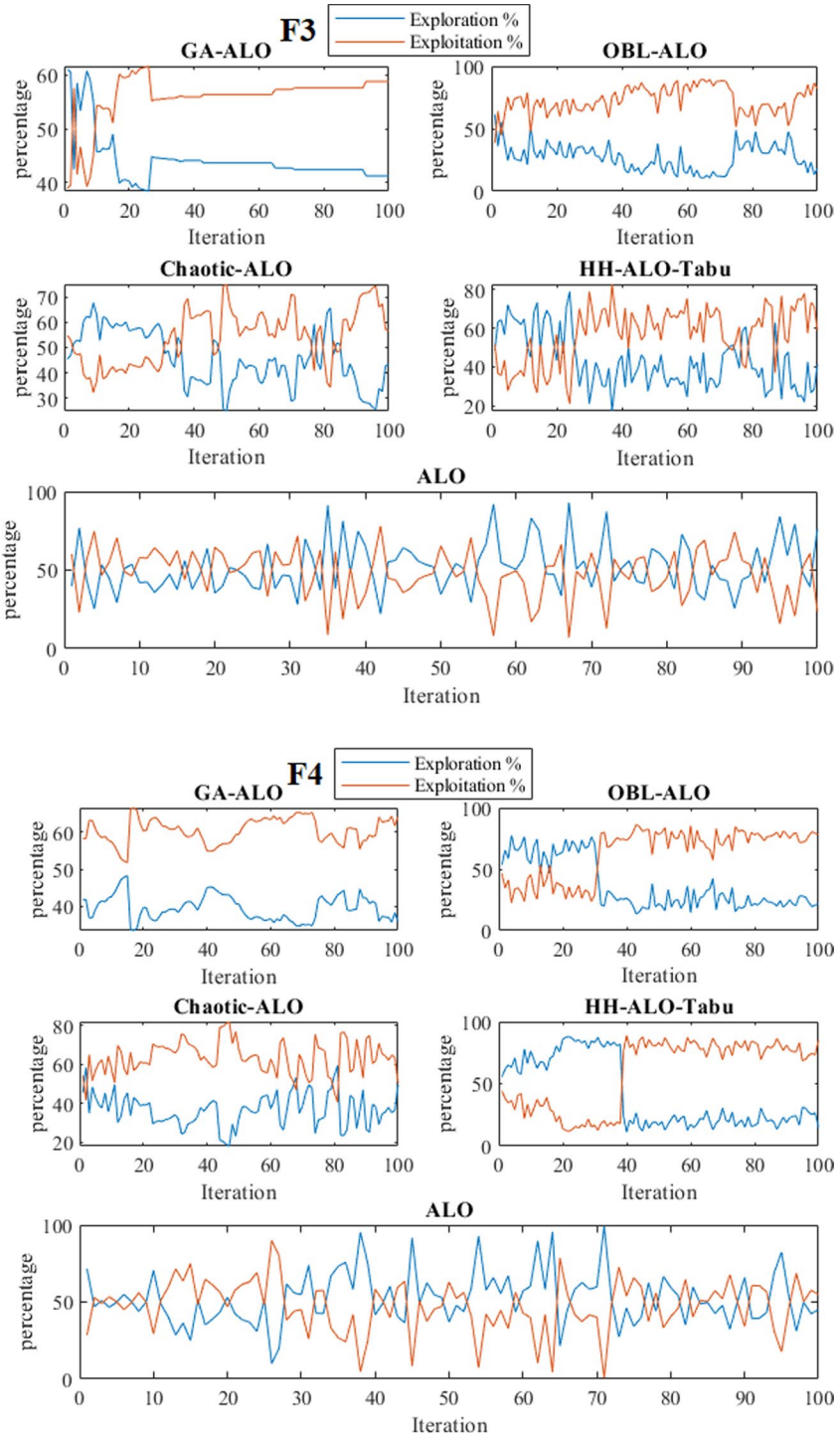

Fig. 42 (continued)

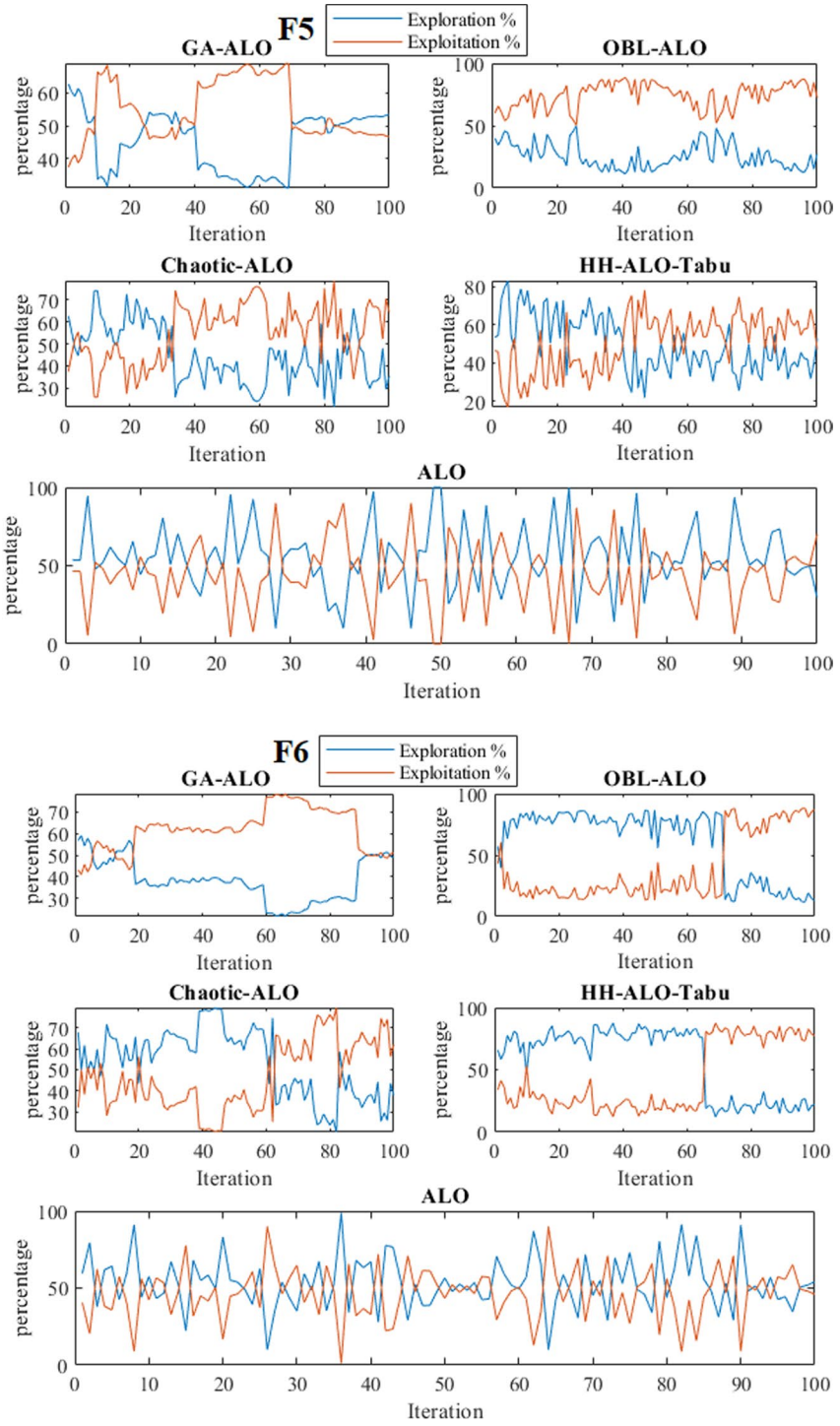

Fig. 42 (continued)

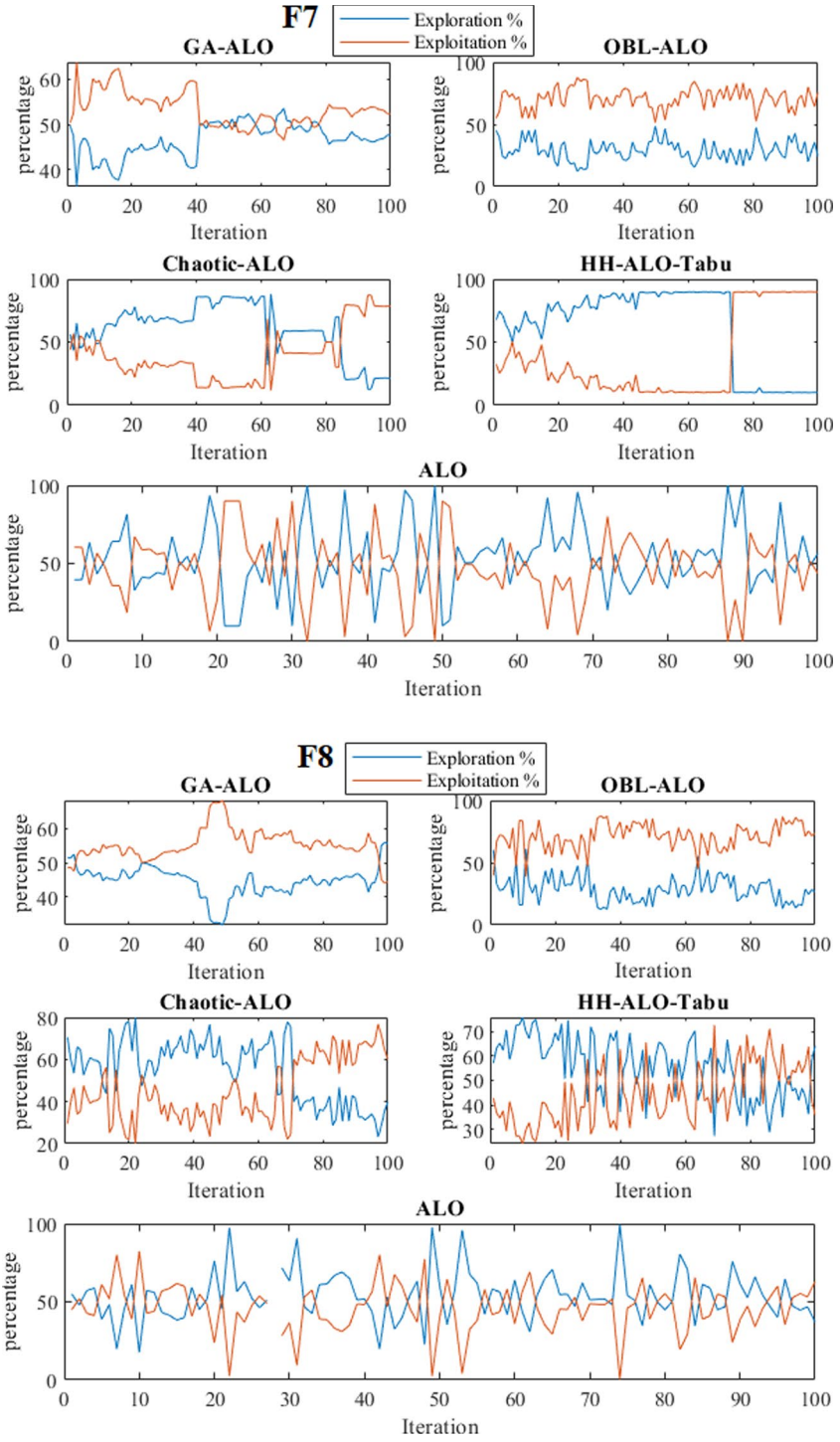

Fig. 42 (continued)

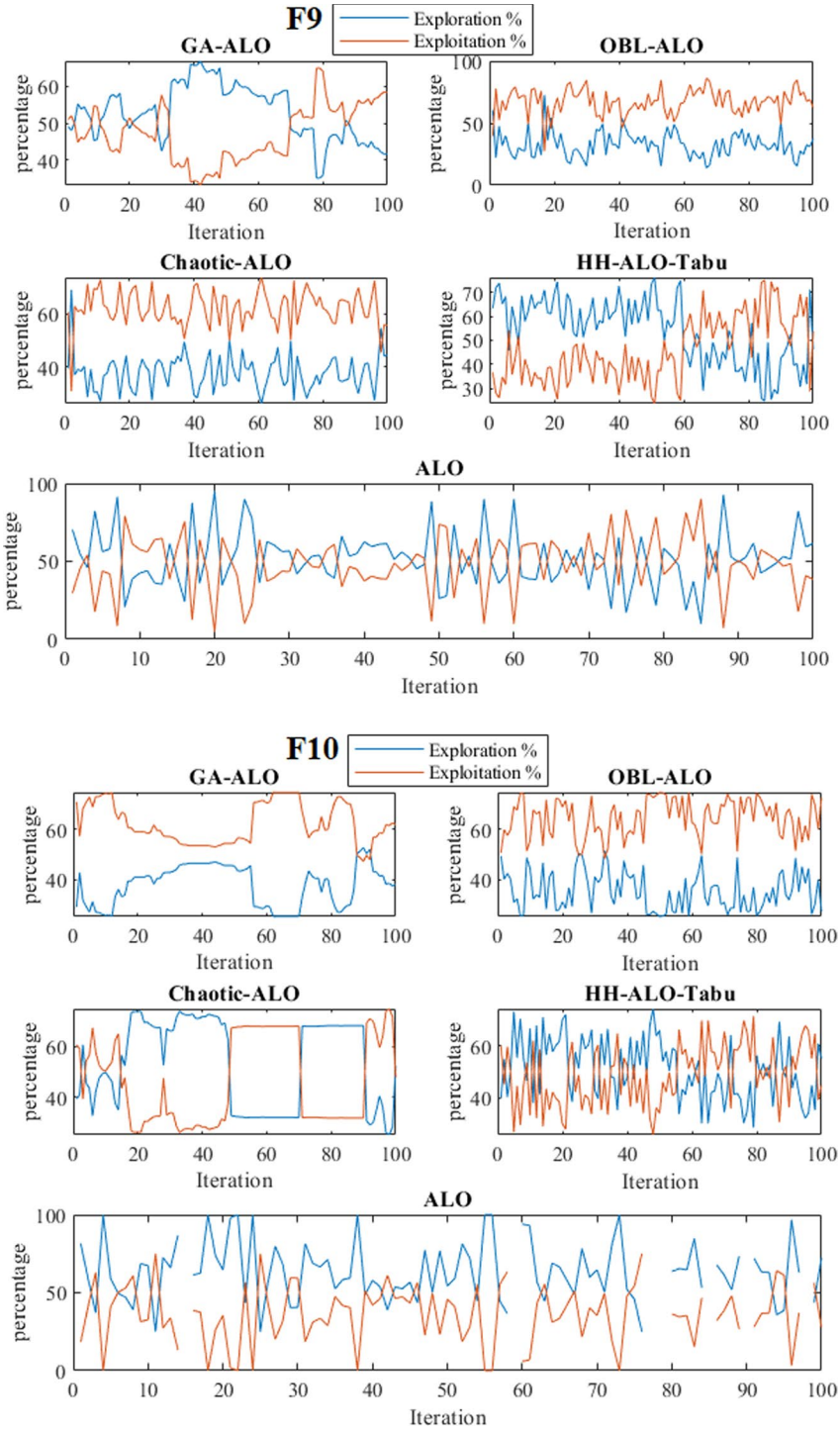

Fig. 42 (continued)

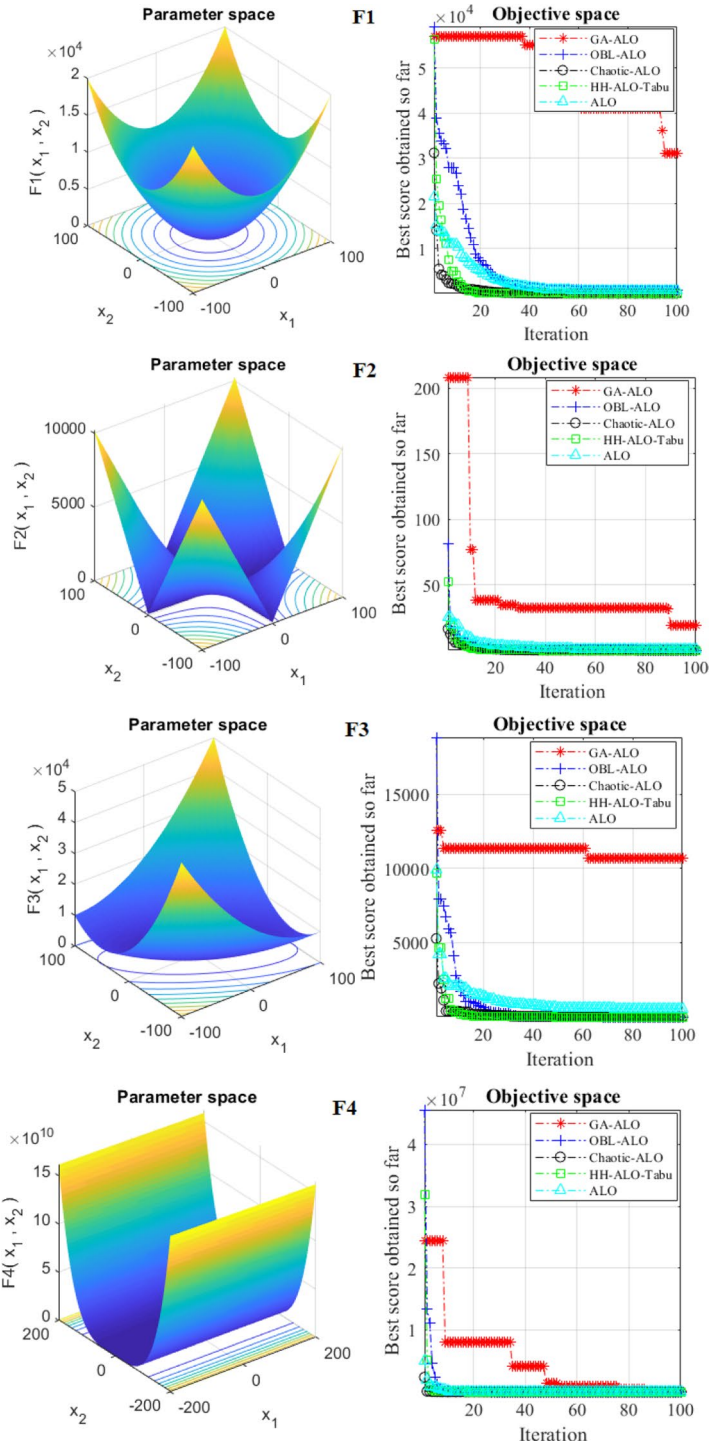

Fig. 43 Convergence curve of the best solution during optimization of CEC 2005 functions

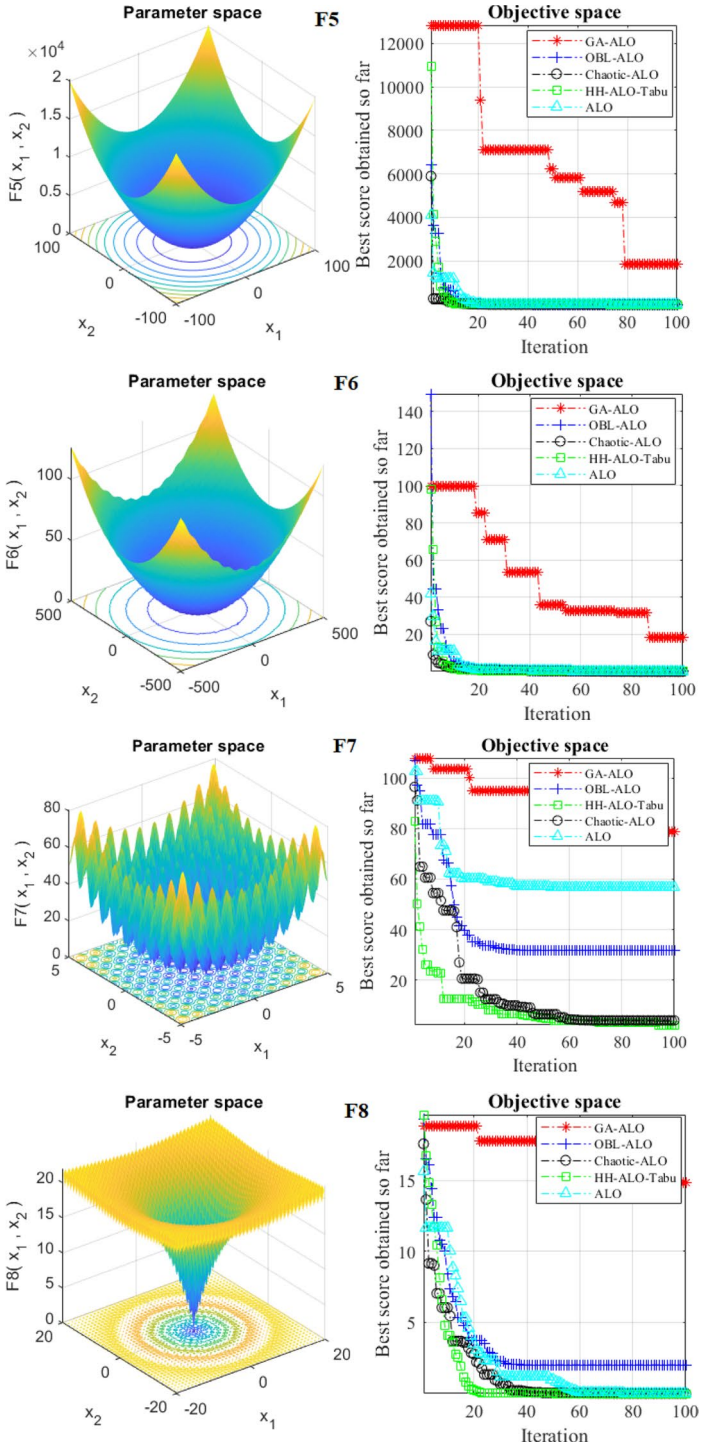

Fig. 43 (continued)

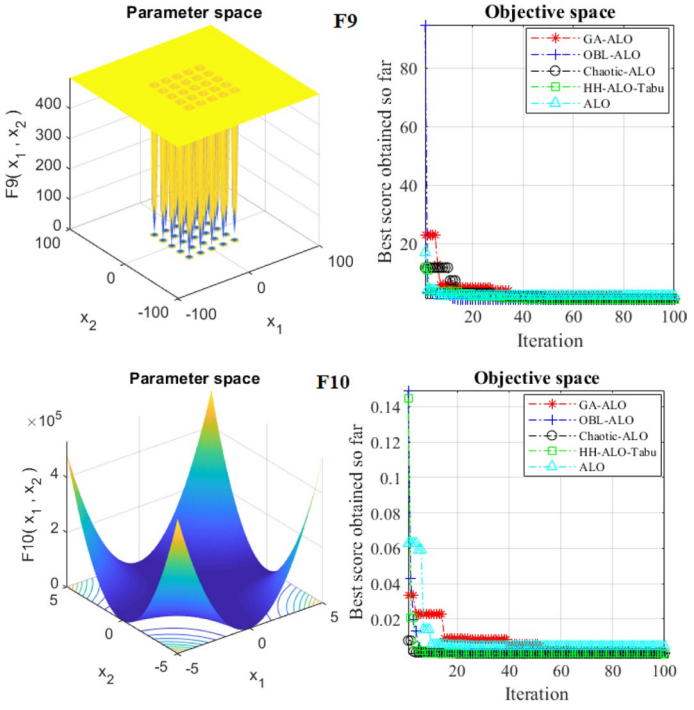

Fig. 43 (continued)

In Fig. 44, we can see that the value of average fitness for all agents has been declining during iterations. The average fitness value of GA-ALO and ALO for 7 out of 10 functions with a lower slope compared to other algorithms is decreased. For a given optimization problem, the proposed HH-ALO-Tabu algorithm can improve the fitness of initial random solutions. One reason for this superiority of the proposed method compared to standard ALO is using a new updating strategy (i.e., relocation) for ant lions. The standard ALO performs the updating process for an ant lion only when a better ant for that ant lion is found. The proposed method updates (relocates) each ant lion based on three parameters (i.e., iteration number, the fitness of the corresponding ant, and elite ant lion), so ant lions move toward better ants and also ant lions over time.

6.2.3 Comparison with stat-of-the-art methods

In general, an optimization technique should be capable of exploring the search space, exploiting promising regions, and convergent on the best solution. In this section, four state-of-the-art algorithms (SSA, GWO, WOA, and MVO) are compared.

With the help of a modern benchmark function (CEC 2019), the proposed method's convergence behavior is analyzed. At the beginning of the optimization process, sharp changes are visible, but when exploitation takes place at the end, these sharp changes are reduced. The HH-ALO-TAbu algorithm's convergence behavior in CEC 2019 benchmark functions is studied by looking at the search history, the trajectory of the

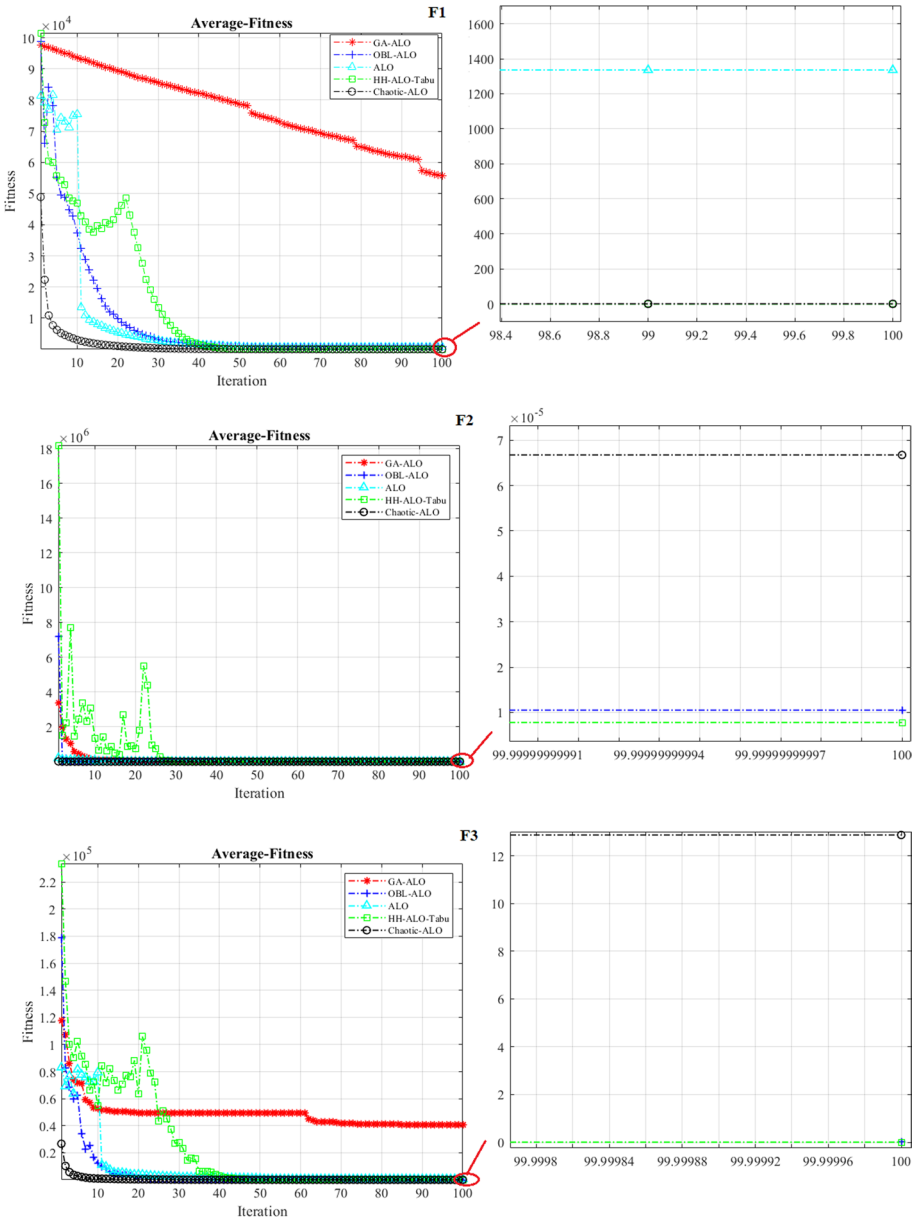

Fig. 44 Average fitness of search agents during optimization of CEC 2005 functions

first search agent, the average fitness, exploration vs. exploitation percentage, and convergence curve are presented in Figs. 45, 46, 47, 48 and 49.

Over 500 iterations, Fig. 45 shows the location of the best agents for different methods. In most functions, the spread of points can be recognized and the search agents of methods explore the promising areas and then exploit the best position. A suitable

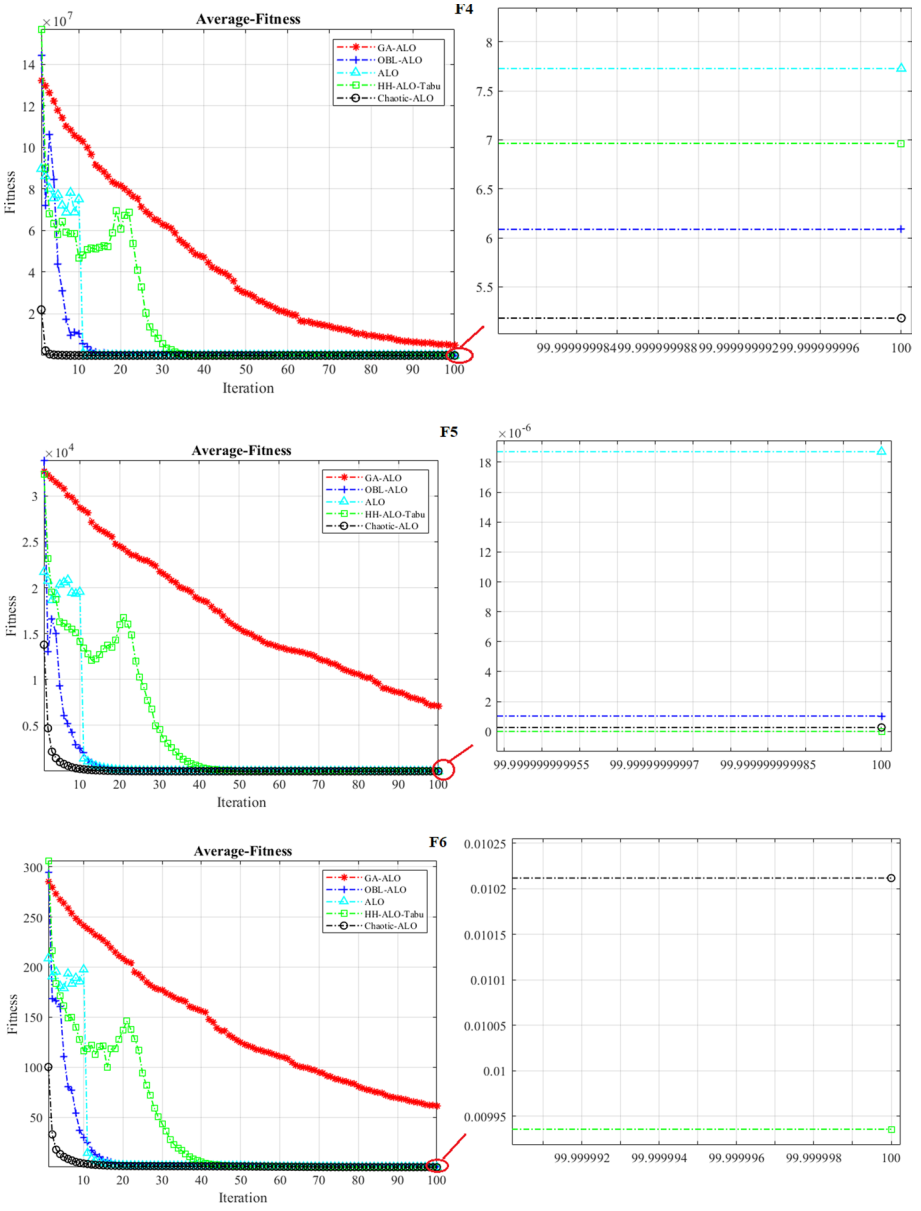

Fig. 44 (continued)

optimization technique should not converge prematurely local optimal and must be able to escape from it. In *cec*1 and *cec*2, most agents of methods are converged to the same point and it seems these functions have one global optimal. In *cec*5, *cec*6, *cec*8, and *cec*10 which have several local optimal, methods tend to explore search space more than those functions with one global optimal. As a result, the distribution of agents in these functions with this shape is logical.

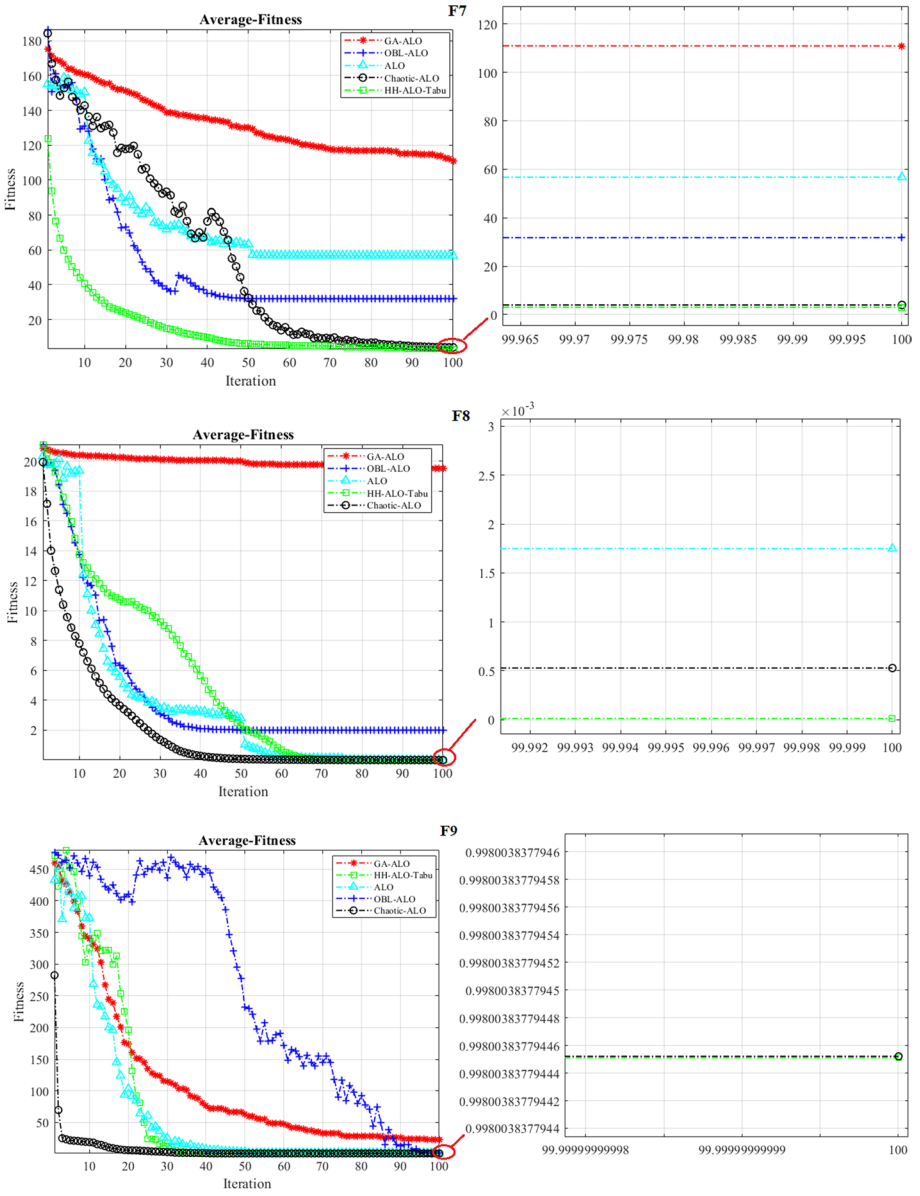

Fig. 44 (continued)

As opposed to WOA and GWO algorithms, which use one parameter to control diversity, HH-ALO-Tabu balances exploration and exploitation by updating the mechanism of ant lions and ants. This dependency on a parameter can be risky since it's highly dependent on the interval of that parameter. In HH-ALO-Tabu, the updating ants in early iterations are performed based on four different spiral paths and the ant's best experience. Then, the ant lions are updated based on the corresponding fitness and positions of

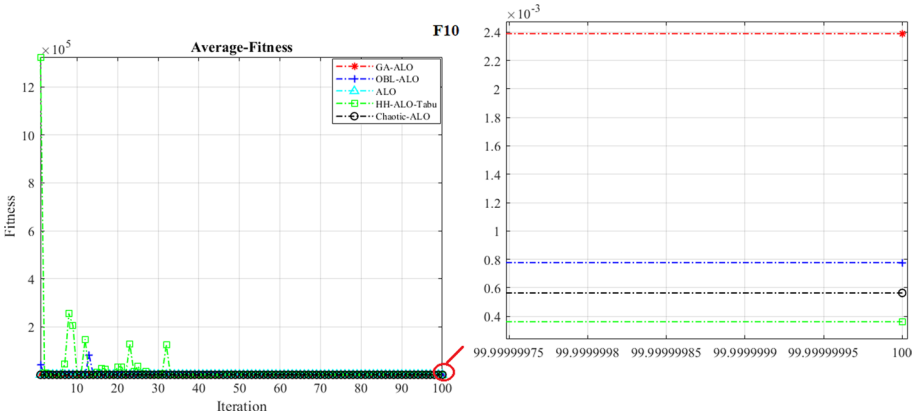

Fig. 44 (continued)

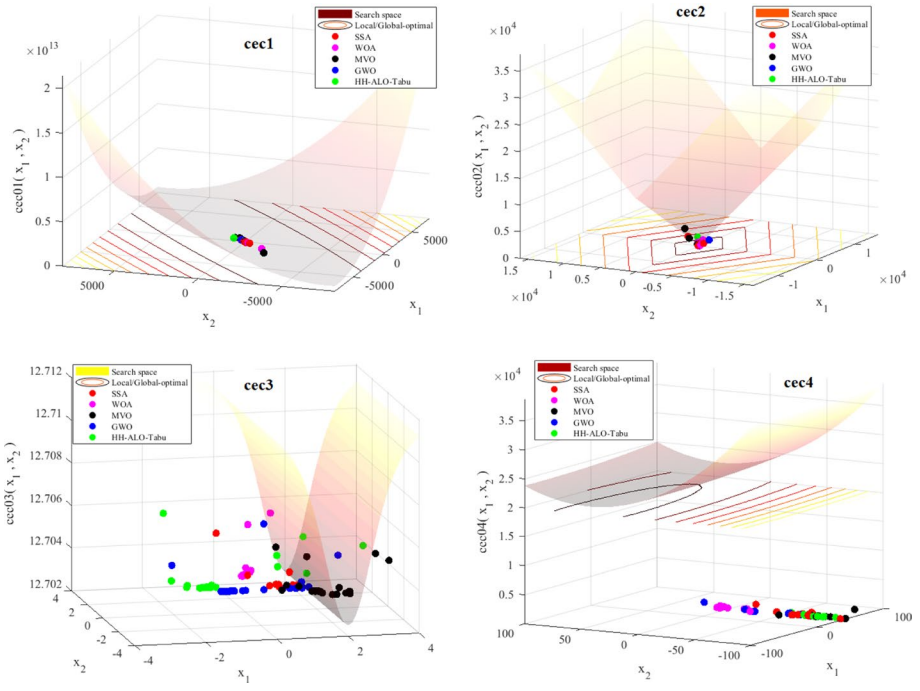

Fig. 45 Search history of agents during solving CEC 2019 functions

ants with a considerable number of iterations. In the last iterations, one spiral path (i.e., the best one) is used to update ants and move them toward the elite ant lion. Moreover, ant lions update their positions based on elite ant lions, and they tend to exploit their neighbors who are elite ant lions.

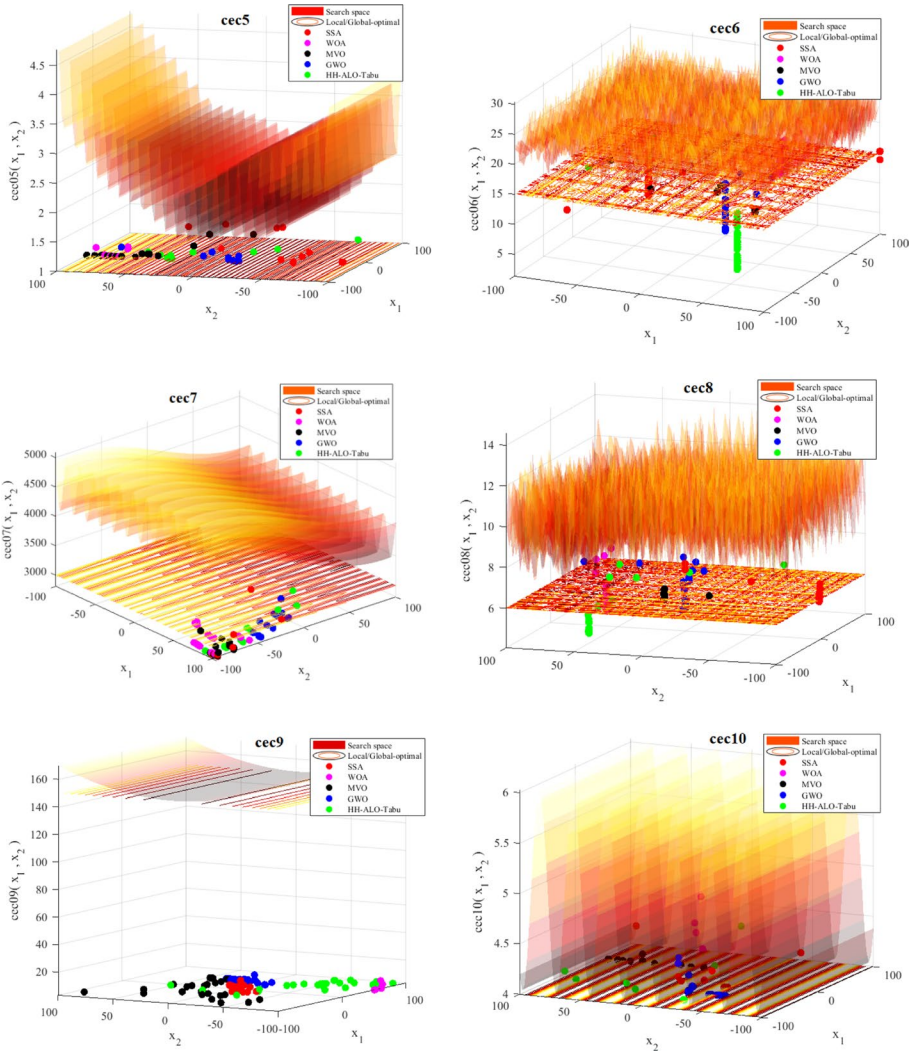

Fig. 45 (continued)

From Fig. 46, it can be seen that the search agents of some methods including HH-ALO-Tabu, WOA, and SSA face abrupt fluctuations in the early steps of optimization more than the other two methods (i.e., GWO and MVO). Throughout iterations, sudden changes gradually diminish. Therefore, the three mentioned methods (i.e., HH-ALO-Tabu, WOA, and SSA) perform the exploration activity in the early iteration while this activity is changed to exploiting the best solution and a local search in the last iteration.

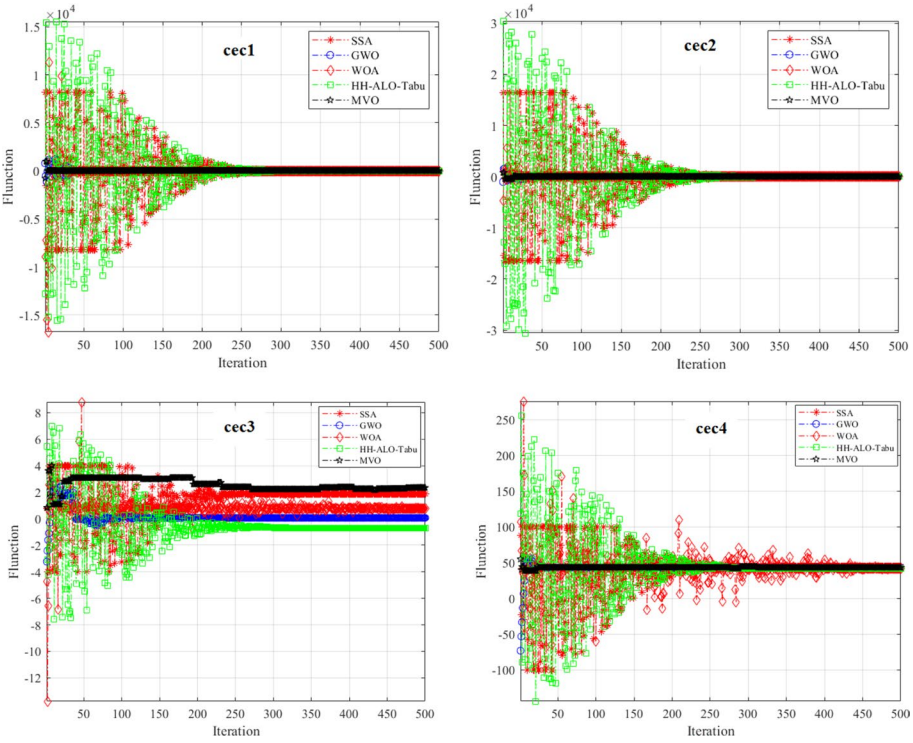

Fig. 46 Trajectory of the first variable in the first search agent (for CEC 2019 functions)

Consider update equations of ant lions [i.e., Eq. (37)] in HH-ALO-Tabu which are designed so that in early iteration the ant lions of $P_{antlion}$ move toward a random ant in P_{ant} with a large step to explore the search space. With increasing the number of iterations, the movement direction of ant lions is changed toward the best solution (i.e., Elite) determined so far and they try to find a better solution. Initially, WOA generates random solutions. A new search agent is chosen randomly each iteration, or a new solution is obtained based on the previous iteration's best solution. Unlike the WOA, the agents of HH-ALO-Tabu are updated so that they move toward a random ant in early iteration and also move toward the best agent with a step search based on their fitness value. Instead of using the best search agent yet found, the position of a search agent is updated based on a random search agent selected during the exploration phase. To maintain a balance between exploration and exploitation, variables have constant ranges that decrease over iterations from 2 to 0.

The movement style of the first agent has been investigated in Fig. 47. For generalization, we must consider the location of each agent at the end of each iteration. As shown in

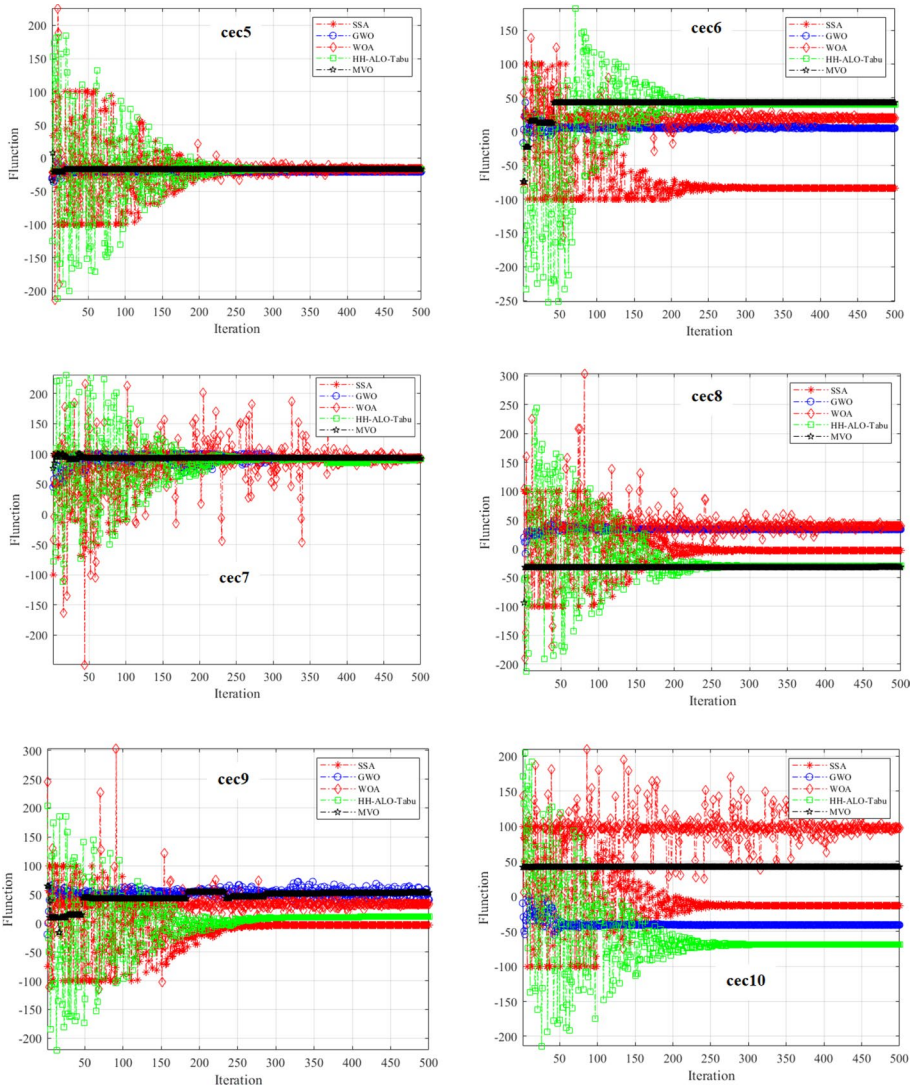

Fig. 46 (continued)

Fig. 47, agents move as a swarm. The result of Fig. 47 can be divided into three groups. SSA and HH-ALO-TAbu form the first group where agents are far from each other and distribute over the search space in early iteration. As the iteration goes up the distance of their agents reduces and they are close to each other. In the last iteration, they tend to converge to a specific location (i.e., solution). In *cec*5, *cec*6, *cec*8, and *cec*10, SSA begins with exploration and then continues with exploitation activity until the last iteration. Exploration is performed in the early stages of HH-ALO-Tabu, followed by exploitation in the middle stages and exploration again in the later stages. In the last iteration, it starts exploitation

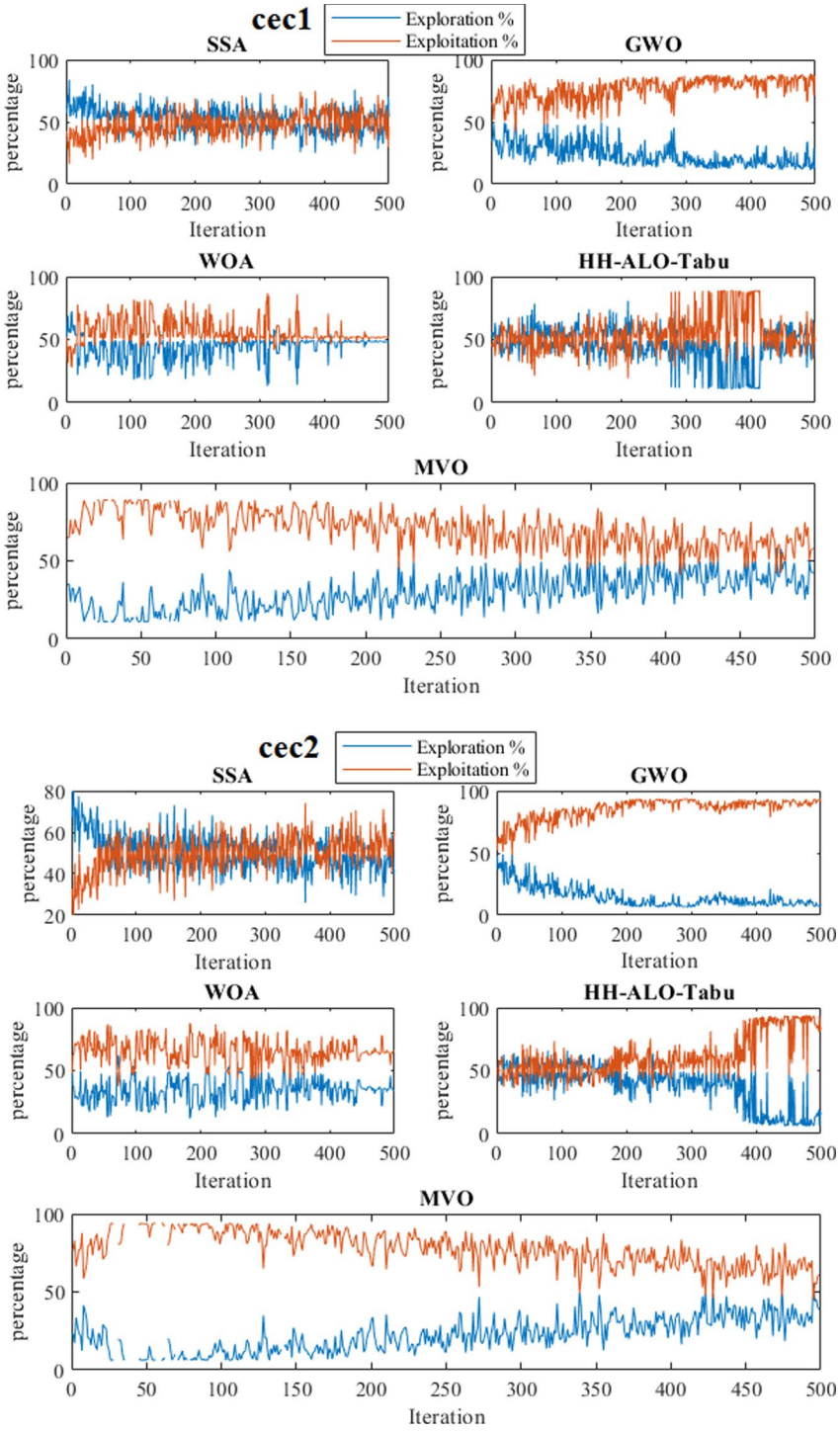

Fig. 47 Exploration vs. exploitation rate (for CEC 2019 functions)

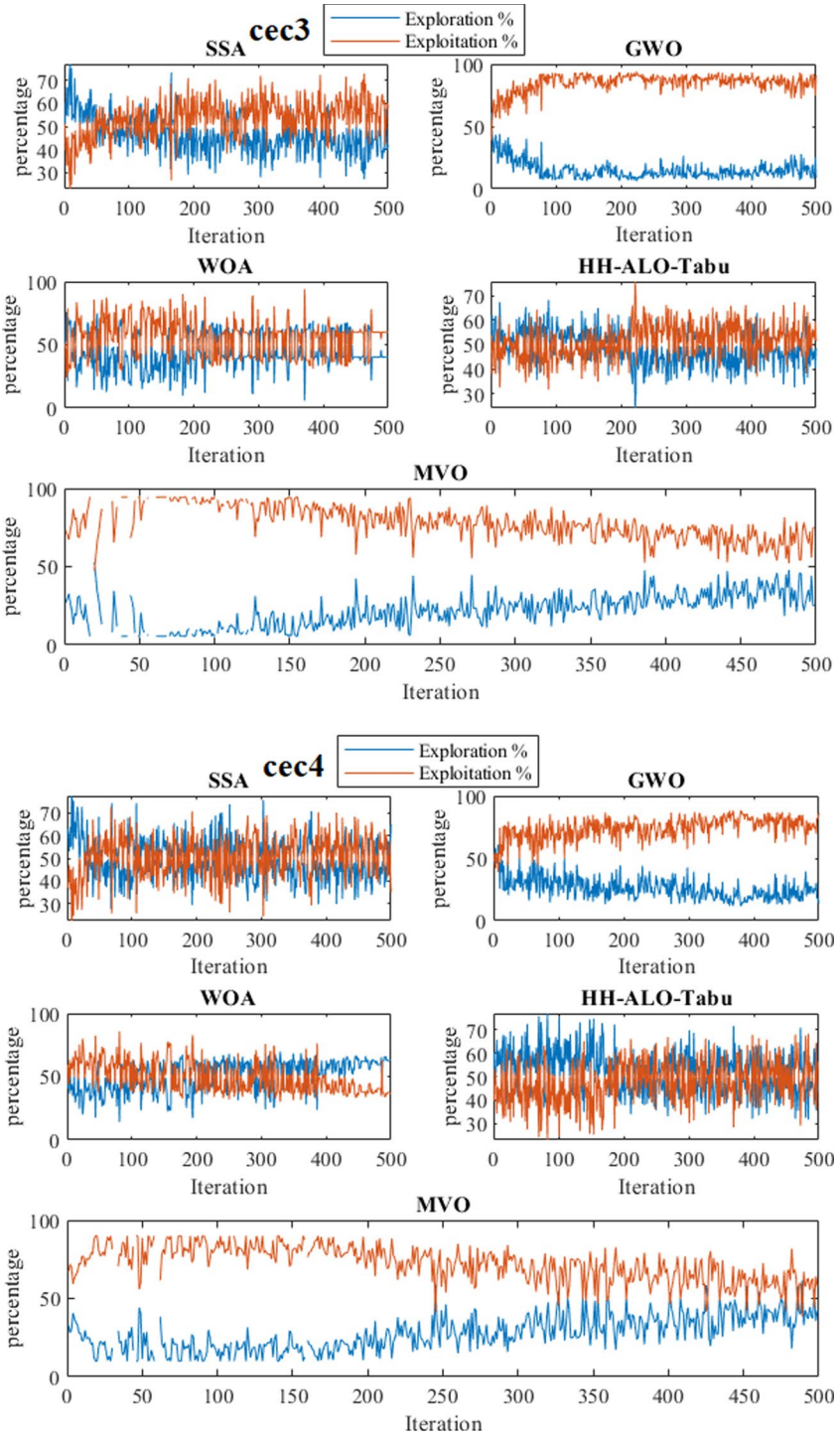

Fig. 47 (continued)

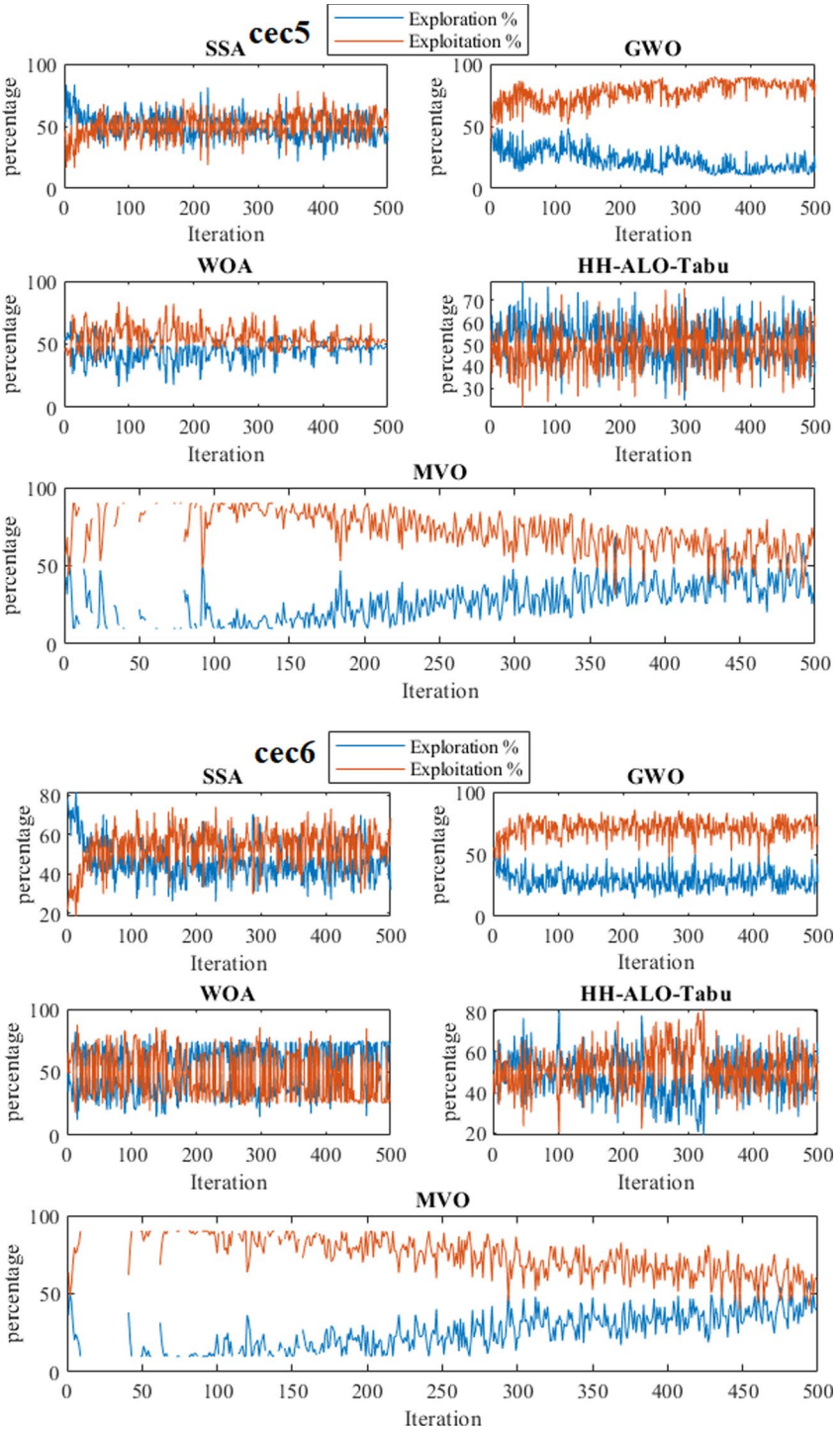

Fig. 47 (continued)

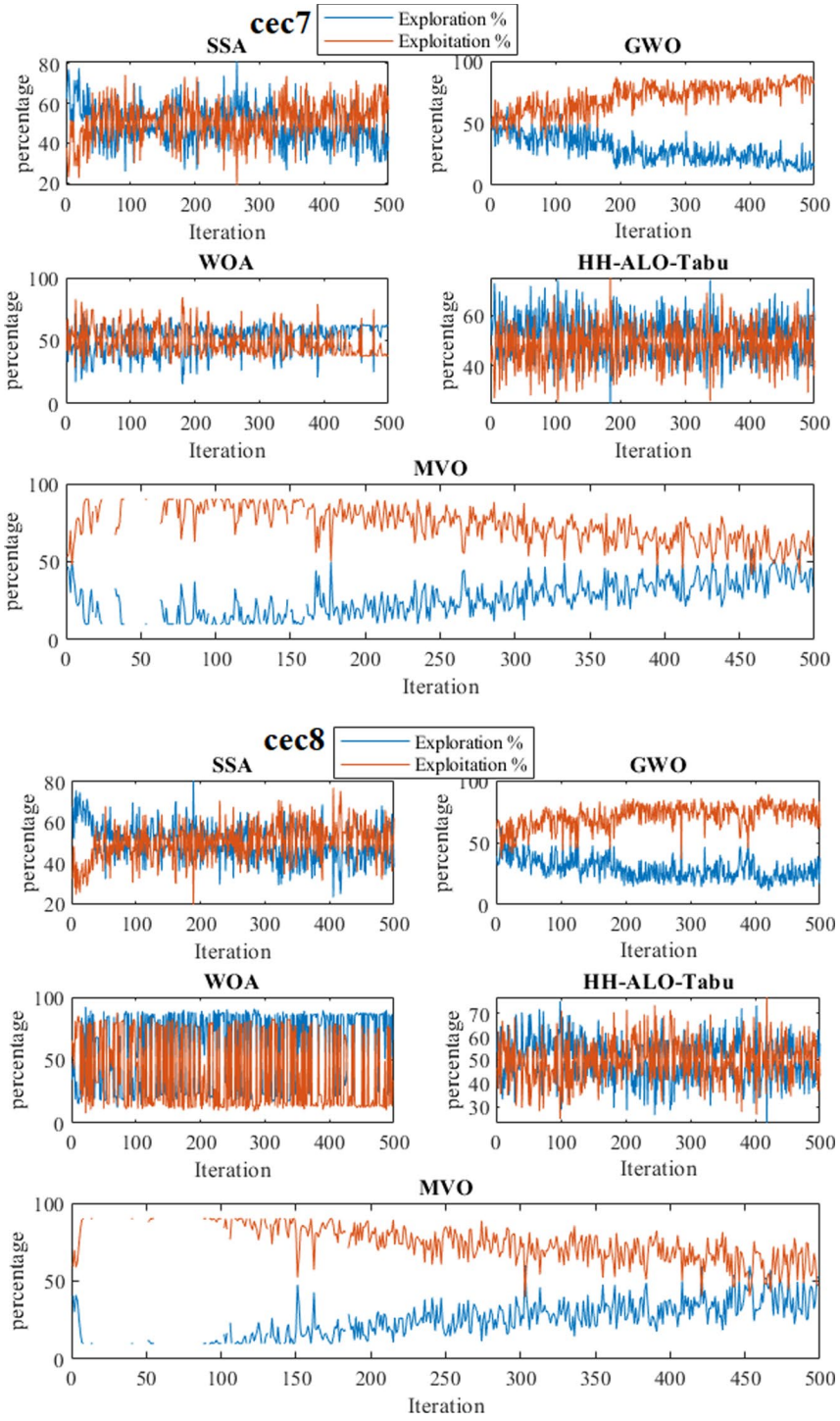

Fig. 47 (continued)

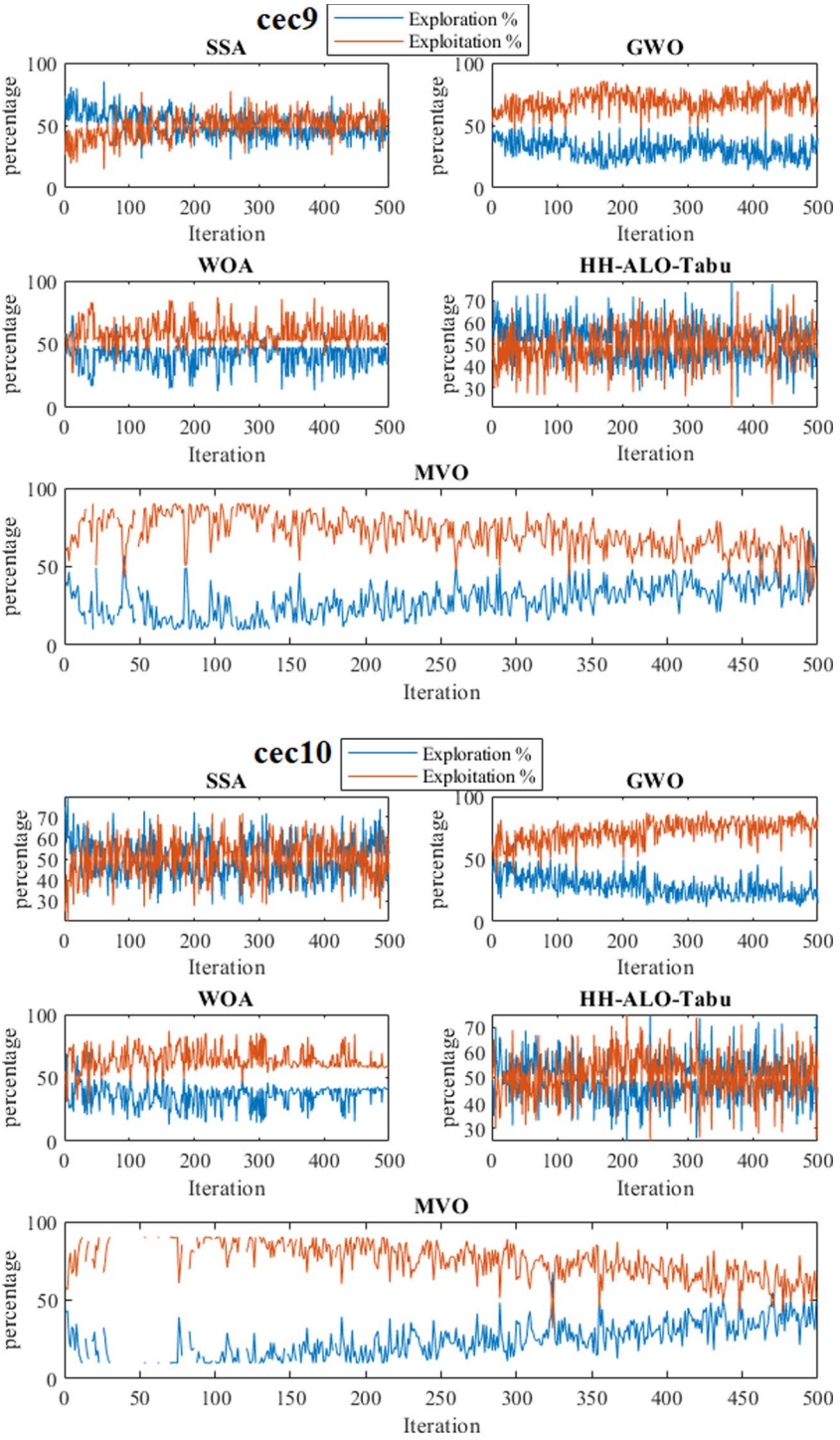

Fig. 47 (continued)

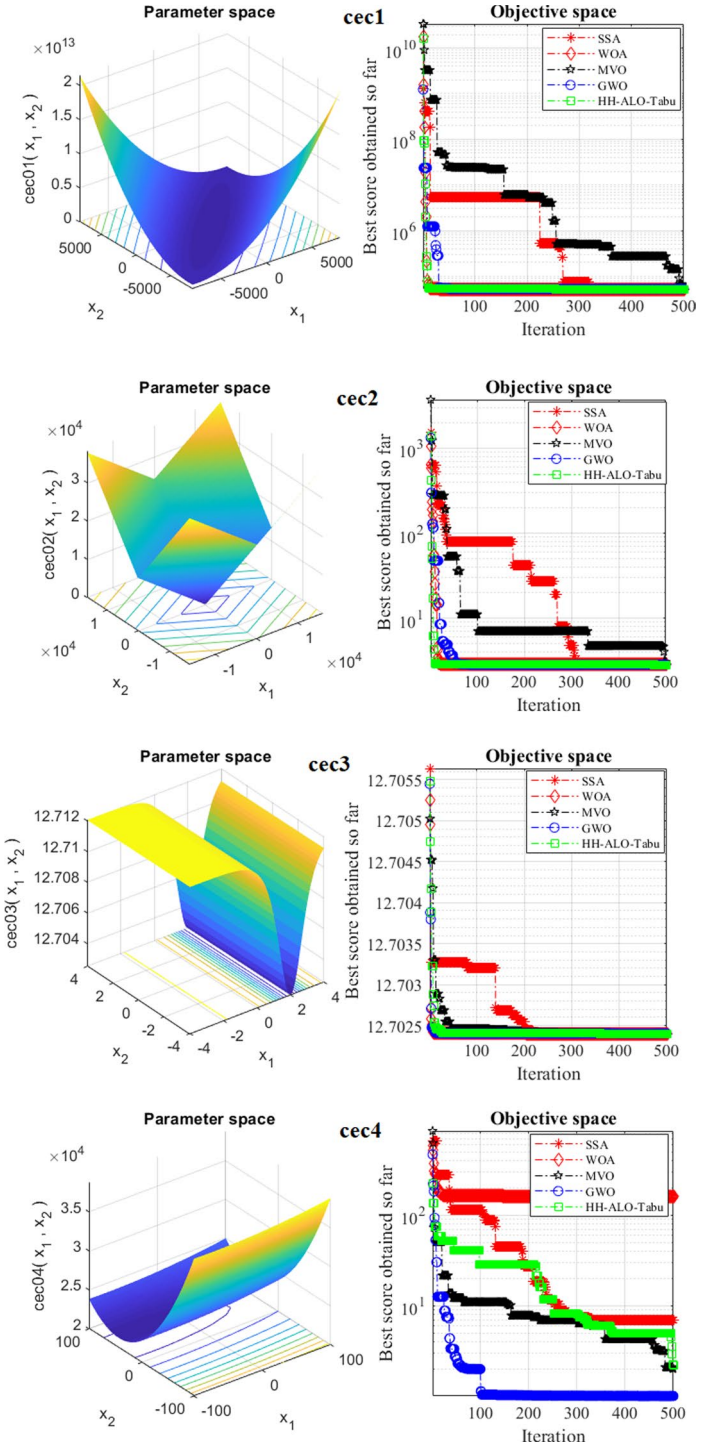

Fig. 48 Convergence curve of the best solution during optimization of CEC 2019 functions

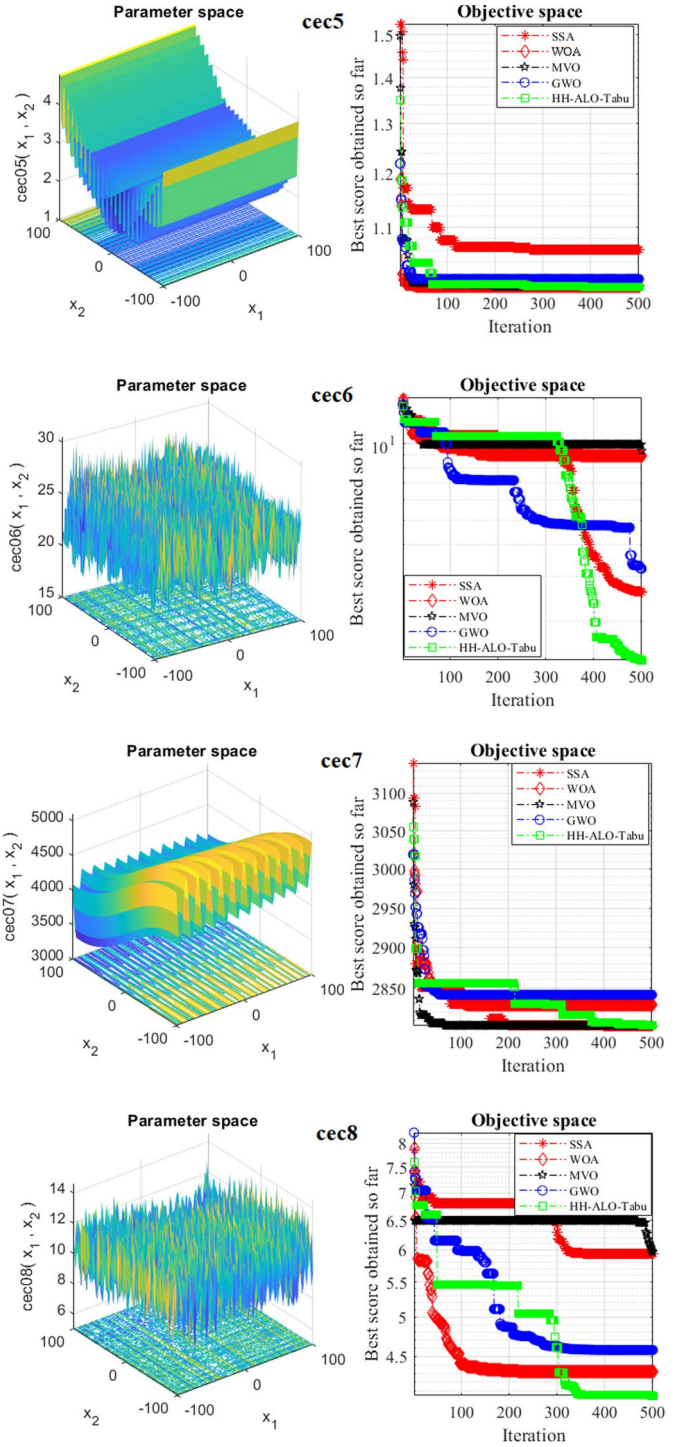

Fig. 48 (continued)

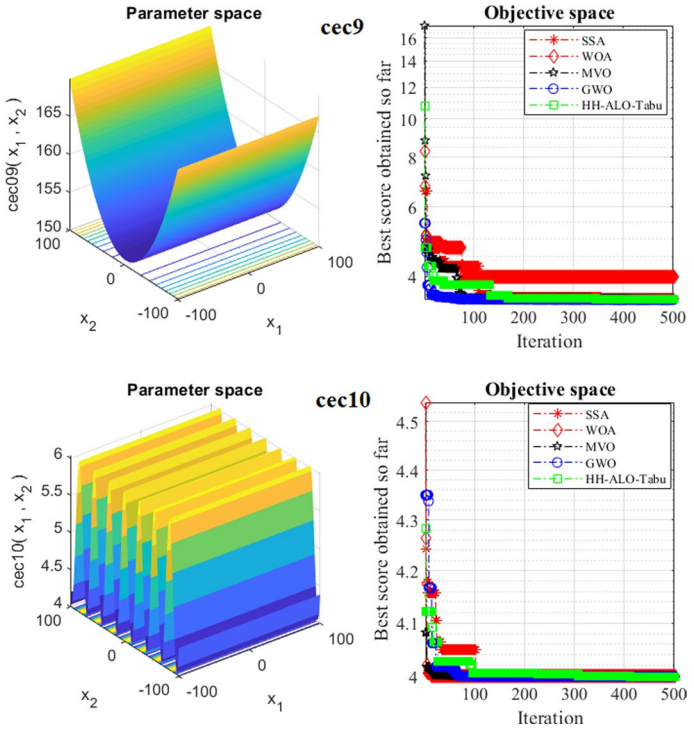

Fig. 48 (continued)

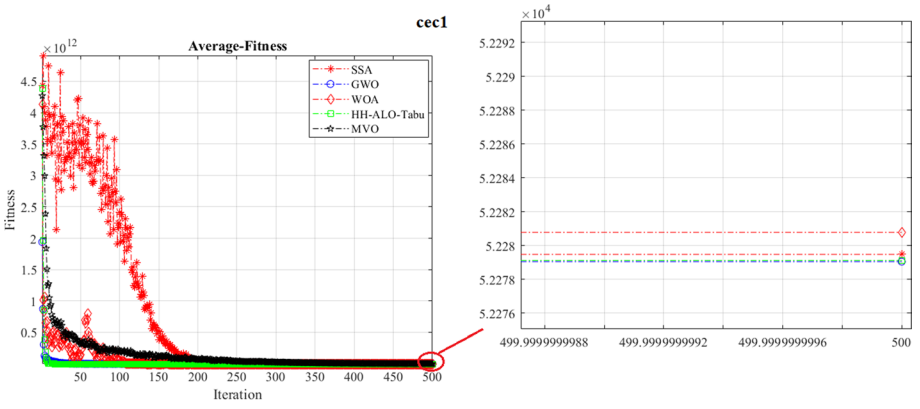

Fig. 49 Average fitness of search agents during optimization of CEC 2019 functions

activity. Consequently, multiple spiral paths are used to update solutions, affecting where agents are located over time.

The second group consists of GWO and MVO. GWO begins its activity with 50% exploration and 50% exploitation in early iterations and then more exploitation activity is

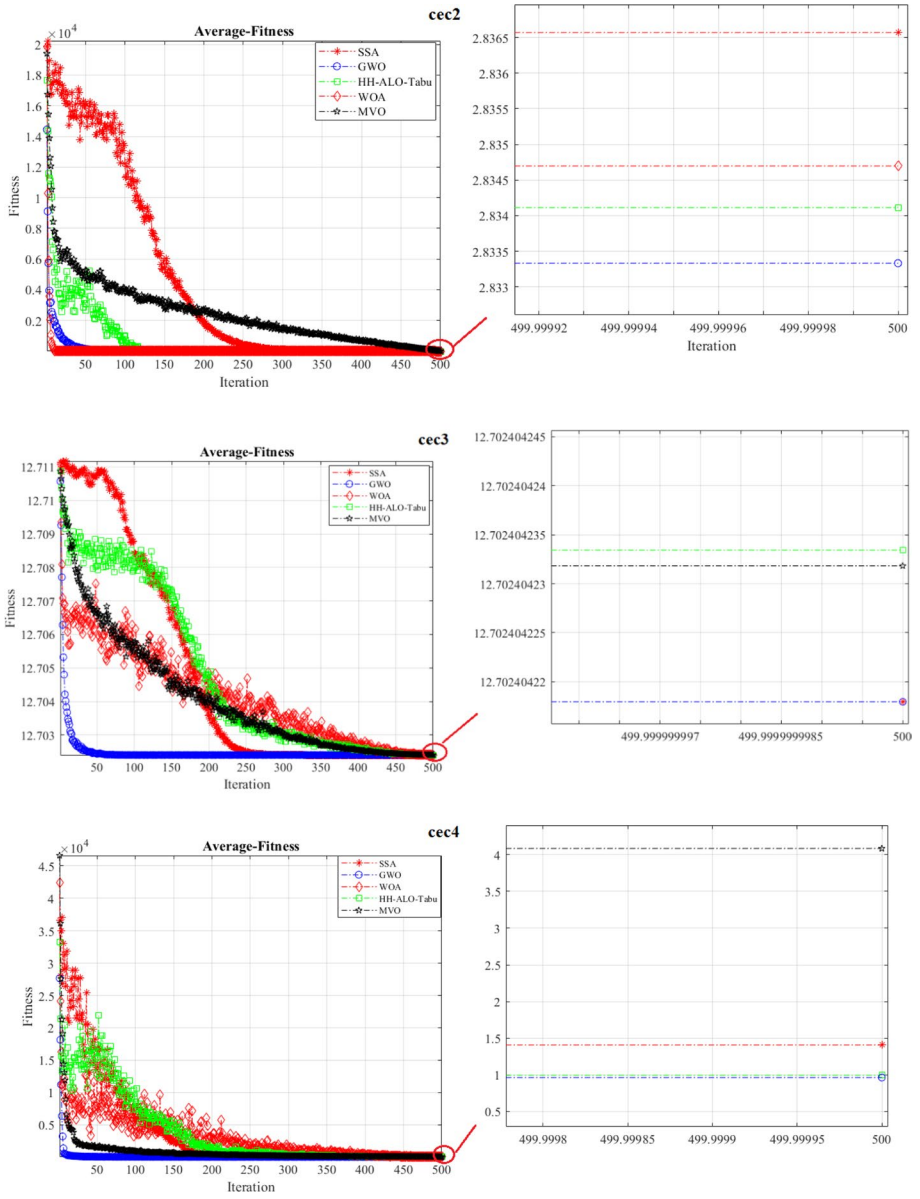

Fig. 49 (continued)

performed. This is contrariwise for MVO, where it begins its work with exploitation and exploration of over 85% and near 15%, respectively. As the iteration increases, the percentage of exploitation activity reduces and the algorithm reaches an equilibrium state where both exploration and exploitation activities of MVO are near 50%. It seems that MVO has a better performance on unimodal functions than multimodal functions because it tends

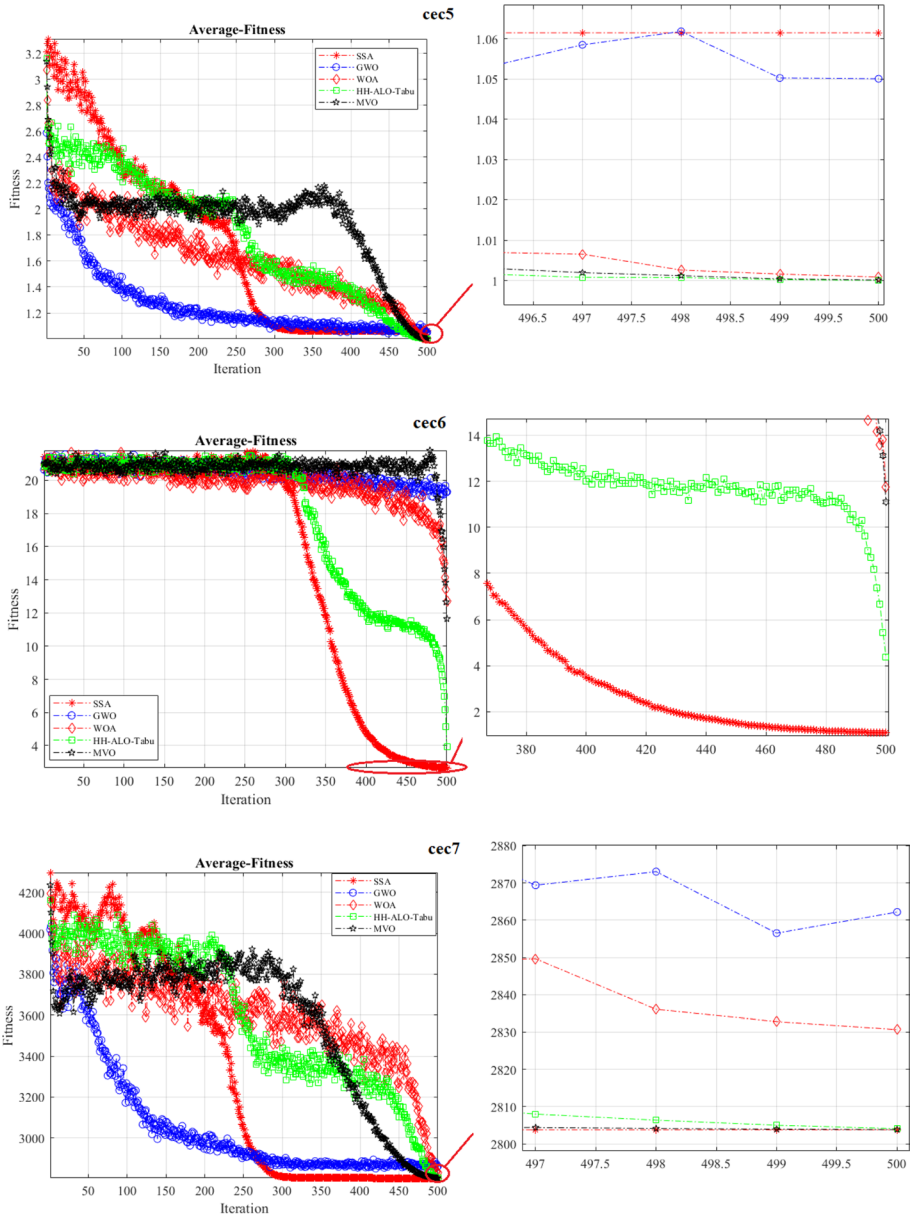

Fig. 49 (continued)

to perform exploitation activities more than exploration over 50% of the time. The multi-modal performance of GWO is better than that of MVO.

Since WOA differs from the other methods in its behavior, it is placed in the third group. In some functions (i.e., *cec*6 and *cec*8) WOA algorithm begins so that its agents are gathered in one area and have a short distance from each other but gradually their distance from each other increases and spread all over the search space. In some functions (i.e., *cec*9

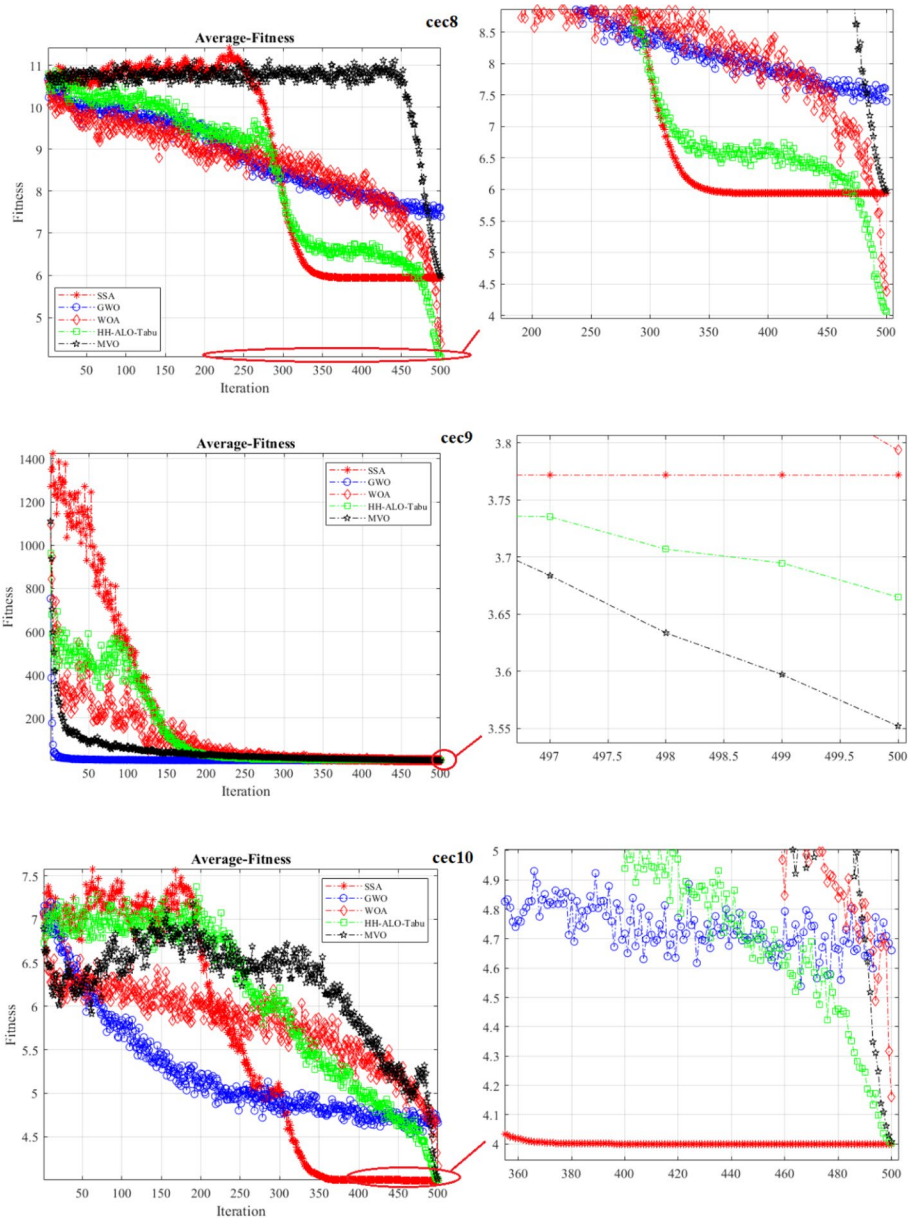

Fig. 49 (continued)

and *cec*10) WOA begins with agents where their distances are far from each other and this condition continues until the last iteration. There is no global optimal in *cec*9 and *cec*10.

Based on Figs. 48 and 49, HH-ALO-Tabu is capable of exploring the search space extensively and identifying the most promising area in fewer iterations than others because

Table 12 Comparative results for CEC 2005 test functions

F	GA-ALO			OBL-ALO			Chaotic-ALO			ALO			HH-ALO-Tabu		
	AVG	STD	Rank	AVG	STD	Rank	AVG	STD	Rank	AVG	STD	Rank	AVG	STD	Rank
F1	1120.2	290.83	5	0.93	0.002	3	0.38	0.05	2	310.5	35.6	4	0.041	0.003	1
F2	44.6	4.78	5	0.34	0.001	2	0.48	0.003	3	8.08	0.23	4	0.005	0.0006	1
F3	8327.5	390.7	5	25.48	1.18	3	18.32	2.35	2	103.8	12.56	4	6.25	0.76	1
F4	33.65	0.57	5	0.15	0.0041	3	0.52	0.038	1	9.75	0.034	4	0.064	0.0007	2
F5	1910.4	305.11	5	1.1	0.0045	3	0.082	0.0031	2	1.5	0.068	4	1	0.32	1
F6	14.87	3.29	5	0.075	0.0081	3	0.0038	0.0006	2	5.38	0.033	4	0.0005	0.00002	1
F7	73.55	4.11	5	27.86	3.79	4	9.53	2.56	2	18.92	0.089	3	8.36	2.11	1
F8	23.04	2.05	5	10.08	0.037	3	0.0047	0.00049	1	18.85	1.076	4	0.0057	0.00028	2
F9	0.156	0.021	4	0.088	0.016	2	0.113	0.093	3	1.75	0.42	5	0.0083	0.00091	1
F10	0.0043	0.0012	3	0.0249	0.013	4	0.0032	0.002	2	0.0355	0.033	5	0.0008	0.007	1
AVG rank	4.7			3			2			4.1			1.2		

Table 13 Comparative results on CEC 2019 test functions

F	SSA			MVO			GWO			WOA			HH-ALO-Tabu		
	AVG	STD	Rank	AVG	STD	Rank	AVG	STD	Rank	AVG	STD	Rank	AVG	STD	Rank
cec1	2.8E+04	2.1E+03	3	8.8E+04	1.5E+04	5	2.2E+04	1.4E+03	2	5.8E+04	1.1E+04	4	1.05E+04	3.2E+03	1
cec2	1.7	0.33	4	4.5	2.3	5	1.5	0.51	3	1.2	0.28	2	1.04	0.0040	1
cec3	17.81	0.49	5	17.064	0.398	4	13.2	0.62	3	13.1	0.073	2	9.8	0.44	1
cec4	8.03	0.75	5	6.82	0.834	3	7.51	0.63	4	1.002	0.001	1	1.04	0.002	2
cec5	1.13	0.044	4	1.16	0.18	5	1.052	0.013	2	1.11	0.045	3	1.027	0.0034	1
cec6	6.18	1.35	5	4.6	2.34	2	4.71	1.72	3	5.12	1.65	4	3.21	0.83	1
cec7	52.19	6.65	4	53.41	7.35	5	35.62	8.27	2	40.06	10.48	3	20.2	3.13	1
cec8	3.93	0.64	5	2.83	0.61	2	3.34	0.58	3	3.83	1.035	4	1.57	0.31	1
cec9	16.57	1.12	5	13.85	1.59	3	16.047	2.37	4	8.05	1.22	2	4.038	0.065	1
cec10	27.58	2.45	5	26.13	2.30	4	24.73	2.39	2	26.11	1.75	3	12.55	1.95	1
AVG rank	4.5			3.8			2.6			2.8			1.2		

Table 14 The *p*-values of Wilcoxon rank-sum test over two benchmark functions

HH-ALO-Tabu vs	GA-ALO		OBL-ALO		Chaotic-ALO		ALO	
	p-value	Sign	p-value	Sign	p-value	Sign	p-value	Sign
F1	0.0082	+	0.0051	+	3.71E−02	+	8.81E−05	+
F2	0.0356	+	0.0349	+	0.233	−	3.56E−03	+
F3	0.0081	+	0.0081	+	4.76E−05	+	0.0051	+
F4	0.00018	+	0.0018	+	2.81E−04	+	0.0217	+
F5	0.00054	+	0.0678	−	0.0338	+	4.36E−08	+
F6	0.01780	+	0.0374	+	2.65E−04	+	0.00074	+
F7	4.31E−06	+	5.55E−04	+	1.42E−05	+	0.045	+
F8	1.65E−04	+	2.98E−06	+	0.188	−	2.65E−04	+
F9	3.55E−02	+	0.0032	+	7.13E−03	+	9.65E−03	+
F10	0.00018	+	0.00018	+	3.89E−05	+	0.00081	+

HH-ALO-Tabu vs	SSA		MVO		GWO		WOA	
	p-value	Sign	p-value	Sign	p-value	Sign	p-value	Sign
cec1	1.65E−04	+	2.35E−05	+	0.0381	+	2.05E−03	+
cec2	3.54E−05	+	3.11E−04	+	1.00E+00	=	0.0328	+
cec3	0.0368	+	0.0456	+	0.00054	+	4.32E−04	+
cec4	4.13E−05	+	3.85E−03	+	8.14E−04	+	0.00081	+
cec5	3.18E−04	+	1.73E−04	+	1.12E−05	+	0.00081	+
cec6	0.00018	+	2.65E−06	+	2.72E−08	+	0.286	−
cec7	3.44E−05	+	0.0396	+	1.53E−04	+	2.88E−04	+
cec8	1.66E−04	+	4.5E−06	+	3.28E−03	+	3.91E−05	+
cec9	2.54E−05	+	1.77E−08	+	3.33E−04	+	1.88E−08	+
cec10	0.0036	+	0.00081	+	0.0891	−	5.11E−04	+

it can generate an initial population near the global optimal with OBL strategy and chaotic function. While other methods generate the first population by random solutions.

6.2.4 The quantitative measure

Quantitative measures show the effectiveness of methods more clearly than qualitative measures, so we perform some qualitative measures as well. In addition, the proposed method performs very well compared to Chaotic-ALO, OBL-ALO, and GWO algorithms. These results prove that the HH-ALO-Tabu algorithm has merit in terms of exploration. Tables 12 and 13 represent the results of the non-parametric Friedman test (Alcalá-Fdez et al. 2009) used in this study to rank the average performance of CEC 2005 and CEC 2019 benchmark functions, respectively. In Tables 12 and 13, the proposed method can obtain the highest ranking by the Friedman test. HH-ALO-Tabu method outperforms for eight functions and nine functions of CEC 2005 and CEC 2019 benchmark functions, respectively. We know that the unimodal functions ($F1$–5) are appropriate for evaluating exploitation. Consequently, experiments indicate the superior performance of the proposed method in exploiting the optimum. While multimodal functions ($F6$–10 and $cec1$–10) have many

Table 15 Impact of different improvements on CEC 2005 benchmarks

Functions (2005)	SOC-ALO		OBL-ALO		Chaotic-ALO		ALO	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1	0.30	0.0045	0.93	0.002	0.38	0.05	310.5	35.6
F2	0.32	0.018	0.34	0.001	0.48	0.003	8.08	0.23
F3	13.65	3.28	25.48	1.18	18.32	2.35	103.8	12.56
F4	0.13	0.0053	0.15	0.0041	0.52	0.038	9.75	0.034
F5	1.19	0.21	1.1	0.0045	0.082	0.0031	1.5	0.068
F6	0.081	0.003	0.075	0.0081	0.0038	0.0006	5.38	0.033
F7	9.32	2.80	27.86	3.79	9.53	2.56	18.92	0.089
F8	0.051	0.0045	10.08	0.037	0.0047	0.00049	18.85	1.076
F9	0.093	0.0041	0.088	0.016	0.113	0.093	1.75	0.42
F10	0.002	0.0076	0.0249	0.013	0.0032	0.002	0.0355	0.033
Functions (2005)	Tabu-ALO		HH-ALO-Tabu		HH-ALO		IMAL-ALO	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1	2.47	0.89	0.041	0.003	0.24	0.085	1.52	0.093
F2	0.86	0.024	0.005	0.0006	0.30	0.002	0.59	0.41
F3	20.06	2.85	6.25	0.76	10.17	1.084	19.27	3.18
F4	0.9531	0.0854	0.064	0.0007	0.09	0.0083	0.8002	0.0039
F5	1.27	0.73	1	0.32	1.09	0.09	1.25	0.12
F6	0.1045	0.0025	0.0005	0.00002	0.007	0.0029	0.0053	0.0007
F7	12.83	3.19	8.36	2.11	9.005	2.49	13.05	1.54
F8	1.42	0.81	0.0057	0.00028	0.057	0.0033	0.7730	0.13
F9	1.2049	0.093	0.0083	0.00091	0.084	0.0031	1.0053	0.6987
F10	0.0290	0.0026	0.0008	0.007	0.002	0.0008	0.0306	0.0002

local optima and so are appropriate for evaluating the exploration ability of an algorithm. According to Tables 12 and 13, the HH-ALO-Tabu algorithm outperforms GA-ALO and ALO for all of the multimodal functions in CEC 2005 and outperforms SSA and MVO for all of the multimodal functions in CEC 2019.

Algorithm performance can only be evaluated using the mean and STD metrics. However, statistical tests based on the outcomes of each execution can prove that the outcomes are statistically significant. To obtain p-values, we apply the Wilcoxon rank-sum test (García et al. 2010). If the *p*-value is lower than 0.05 then there is significant statistical superiority in outcomes. In Table 14, the *p*-values illustrate the statistical superiority of the proposed strategy. The symbols of "+", "=", and "−" represent that HH-ALO-Tabu is superior to, equal to, and inferior to other methods, respectively. In Table 14, HH-ALO-Tabu is slightly better than GA-ALO, OBL-ALO, Chaotic- ALO, and ALO respectively in 10, 9, 8, and 10 out of 10 CEC 2005 problems. It can be seen that HH-ALO performs better than GWO, WOA, and MVO, respectively in 10, 8, and 9 out of 10 of CEC 2019 problems. As a result, the proposed method is superior to other competing methods.

To demonstrate the effectiveness of the various improvements incorporated into ALO, we present Tables 15 and 16.

Table 16 Impact of different improvement on CEC 2019 benchmarks

Functions (2019)	SOC-ALO		OBL-ALO		Chaotic-ALO		ALO	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
cec1	3.56E+07	1.05E+07	2.56E+08	1.86E+08	2.21E+08	3.04E+07	1.53E+09	3.23E+09
cec2	109.918	17.063	163.428	25.177	128.619	31.204	186.958	39.733
cec3	13.497	0.684	19.275	2.436	14.783	3.553	19.287	2.112
cec4	19.340	2.409	20.754	3.390	24.754	3.779	15.398	1.187
cec5	1.215	0.208	1.415	0.090	1.295	0.089	1.425	0.218
cec6	8.182	1.719	9.732	0.297	9.744	1.253	11.047	1.937
cec7	104.604	20.039	184.427	29.580	139.532	18.904	202.471	43.576
cec8	1.724	0.466	5.054	1.003	6.152	1.948	5.004	0.109
cec9	3.983	0.384	2.997	0.811	5.107	0.403	3.193	0.049
cec10	21.150	5.108	38.763	6.761	30.719	4.190	47.857	4.510
Functions (2019)	Tabu-ALO		HH-ALO-Tabu		HH-ALO		IMAL-ALO	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
cec1	5.16E+07	2.16E+06	1.05E+04	3.2E+03	4.18E+06	1.36E+06	1.22E+07	3.14E+06
cec2	113.971	19.978	1.04	0.0040	53.128	6.648	65.447	12.541
cec3	17.341	2.039	9.8	0.44	11.997	1.759	17.345	2.859
cec4	20.755	2.105	1.04	0.002	9.647	0.851	13.125	2.497
cec5	1.305	0.416	1.027	0.0034	1.200	0.037	1.286	0.375
cec6	8.882	1.896	3.21	0.83	6.862	1.253	7.918	0.688
cec7	112.511	24.618	20.2	3.13	63.514	3.505	87.605	7.392
cec8	3.749	0.246	1.57	0.31	1.700	0.043	3.019	0.183
cec9	4.987	1.973	4.038	0.065	4.013	0.757	3.007	0.046
cec10	27.692	4.858	12.55	1.95	14.311	2.180	23.912	3.733

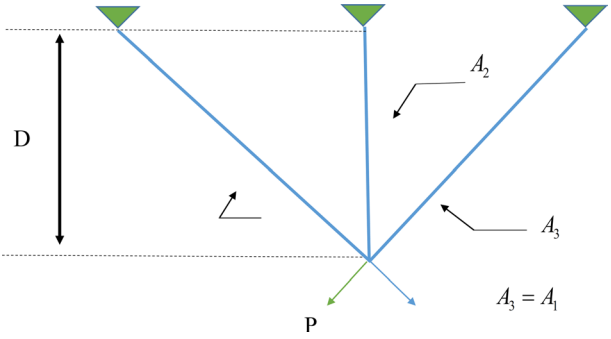

Fig. 50 Model of the three-bar truss (Abualigah et al. 2021b)

We consider 20 benchmark functions (i.e., 5 unimodal and 15 multimodal and complex functions from CEC 2005 and CEC 2019) to evaluate each improvement. The details of each improvement are listed as follows:

- *ALO* the standard ant lion optimizer.
- *Chaotic-ALO* it is ALO where the first population is initialized with the Tent chaotic function.
- *OBL-ALO* it is ALO where the first population is initialized with the standard opposition learning strategy.
- *IMAL-ALO* it is ALO where the ant lions are updated according to Eqs. (37)–(39).
- *SOC-ALO* it is ALO where the first population is initialized with the Tent function. The standard population and ants update their positions with logarithmic spiral movement.
- *Tabu-ALO* it is ALO where the ants update their positions with four spirals (i.e., logarithmic, rose, hypotrochoid, and Archimedes). The Tabu list is used to change the form of movement if the algorithm is not improved after several iterations.
- *HH-ALO* it is a hyper-heuristic version of ALO where DE is used to select appropriate configuration among 10 chaotic functions, 4 spiral movements, and 4 OBL strategies.
- *HH-ALO-Tabu* it is the proposed method.

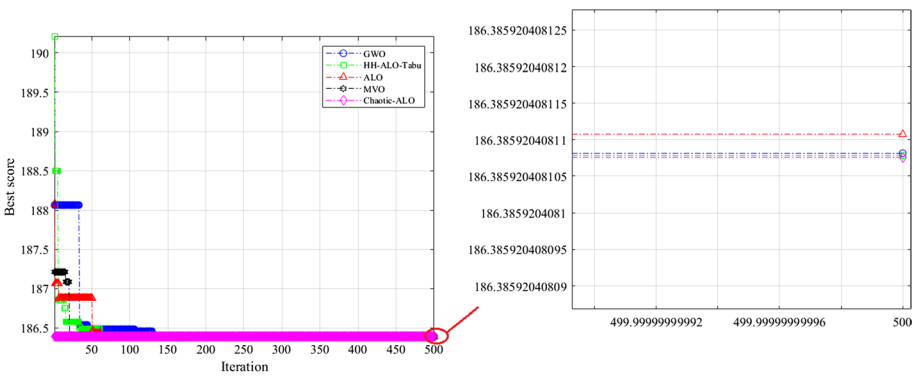

Fig. 51 Convergence curve of different methods for three-bar truss problem

Table 17 The comparison results for the three-bar truss problem

Algorithm	Optimum variables		Optimum weight
	A1	A2	
HH-ALO-Tabu	**0.786848272110945**	**0.288006468605843**	**1.863859204081076E+02**
ALO	0.786848284343577	0.288006432230396	1.863859204081078E+02
MVO	0.786861474942341	0.287972418971453	1.863859205011463E+02
GWO	**0.786848272672500**	**0.288006468359862**	**1.863859204081076E+02**
Chaotic-ALO	**0.786848278538244**	**0.288006460788474**	**1.863859204081076E+02**

The bold refers to the model(s) with optimal cost

Based on the results from CEC 2005, using the Tent function and standard OBL for initializing the population and updating the ant with logarithmic spiral (SCO-ALO) is 38% and 19% better than the situation where the first population only is initialized with standard OBL (i.e., OBL-ALO) or Tent function (i.e., Chaotic-ALO), respectively.

It can be interpreted that only exploitation activity such as updating the ants with spiral movement based on only Elite ant lion (Tabu-ALO) is not appropriate and it achieved a worse result than the HH-ALO-Tabu method that considers exploitation and exploration techniques together.

In Table 16, it can be seen that IMAL-ALO improves the final result by 17% and 29% compared to Tabu-ALO and ALO, respectively. The HH-ALO-Tabu improves the final result by 25% and 38% compared to Tabu-ALO and ALO, respectively. Since they improve the exploration activity of ALO.

The Chaotic-ALO and OBL-ALO have better performance for some functions (e.g., $F1$, $F2$, and $F4$) compared to IMAL-ALO and SCO-ALO. While they show the worst performance for $cec1$, $cec2$ and $cec7$ hence only the appropriate first population doesn't help a meta-heuristic algorithm reach a global optimum.

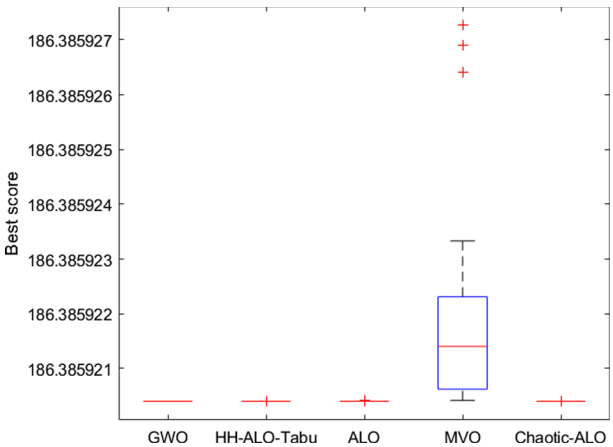

Fig. 52 Box plot for three-bar truss problem

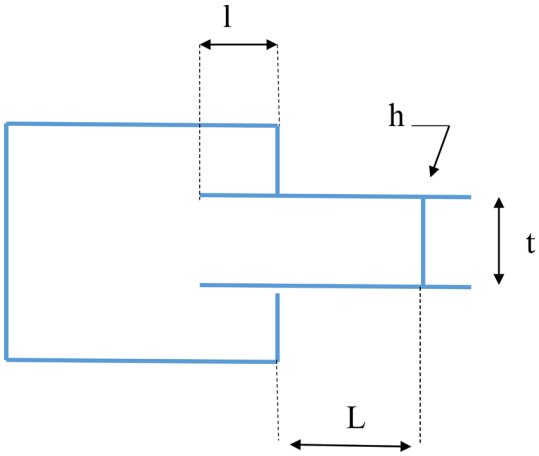

Fig. 53 Model of welded beam design (Abualigah et al. 2021b)

6.2.5 Engineering optimization design

We apply the proposed algorithm to three engineering optimization problems: three-bar truss design, welded beam design, and pressure vessel design. It is necessary to compare HH-ALO-Tabu with four other algorithms to better evaluate its optimization performance (i.e., ALO, MVO, GWO, and Chaotic-ALO). For each problem, the compared algorithms run independently 20 times. The statistical results can be seen as box plots, the convergence of each algorithm is represented throughout iterations and the best result achieved by each algorithm is reported.

A three-bar truss is designed to minimize its weight by minimizing the number of bars. The three-bar truss is illustrated in Fig. 50 as well as its variables. It is necessary to optimize two variables ($A1$ and $A2$). In the three-bar truss problem, the following objectives and constraints are defined (Abualigah et al. 2021b):

$$\text{Consider: } \vec{x} = [x_1\, x_2] = [A_1\, A_2]$$

$$\text{Minimize: } f(\vec{x}) = (2\sqrt{2}x_1 + x_2).l$$

$$\text{Subject to: } g_1(\vec{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}p - \sigma \leq 0$$

$$g_2(\vec{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}p - \sigma \leq 0,$$

$$g_3(\vec{x}) = \frac{1}{\sqrt{2}x_2 + x_1}p - \sigma \leq 0,$$

Variable range: $0 \leq x_1,\ x_2 \leq 1, l = 100$ cm, $p = 2$ kN/cm^2, $\sigma = 2$ kN/cm^2.

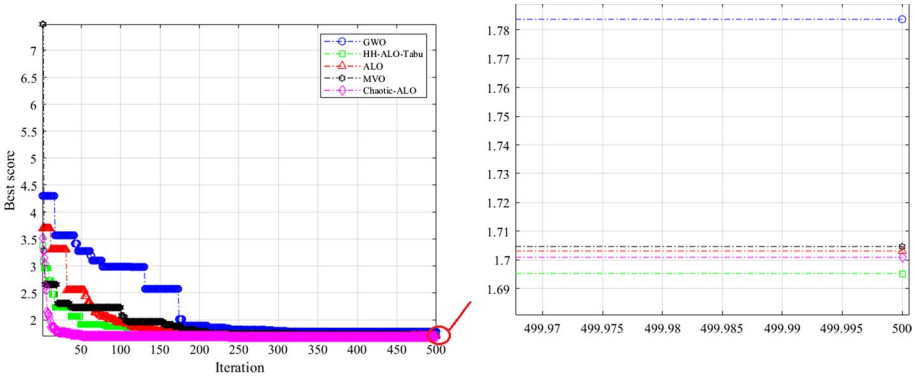

Fig. 54 Convergence curve of the compared methods on the welded beam design problem

Figure 51 shows how compared methods converge after 500 iterations. Almost all methods achieve a similar result, 186.385920.

The optimum value for two variables (i.e., $A1$ and $A2$) along with the final weight for each method can be seen in Table 17. Comparing HH-ALO-Tabu, GWO, and Chaotic-ALO with ALO and MVO, it is obvious that the proposed algorithms have better performance.

Data distributions are generally illustrated using box plots as a standardized method. Figure 52 shows the box plot of different methods after 20 runs. It can be observed that all methods' box plots except MVO are remarkably narrow for the three-bar truss problem. Narrow boxplots suggest that the obtained solutions possessed high levels of agreement with each other. In the three-bar truss problem, solution quality from GWO, Chaotic-ALO, and HH-ALO-Tabu is superior to other algorithms.

Optimal welding costs can be achieved by designing the welded beam. A welded beam problem is illustrated in Fig. 53. Four variables must be optimized (i.e., weld seam thickness h, clamp beam length l, bar height t, and bar thickness b) based on the four constraints shear stress τ, bending stress in the beam Θ, buckling load Pc, and deflection of the beam δ. The objective function and constraints of the welded beam problem are defined as follows (Abualigah et al. 2021b):

$$\text{Consider: } \vec{x} = [x_1\, x_2\, x_3\, x_4] = [h\, l\, t\, b]$$

$$\text{Minimize: } f(\vec{x}) = 1.10471 x_1^2 x_2 + 0.02811 x_3 x_4 (14.0 + x_2)$$

$$\text{Subject to: } g_1(\vec{x}) = \tau(\vec{x}) - \tau_{\max} \leq 0$$

$$g_2(\vec{x}) = \theta(\vec{x}) - \theta_{\max} \leq 0,$$

$$g_3(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \leq 0,$$

$$g_4(\vec{x}) = x_1 - x_4 \leq 0,$$

$$g_5(\vec{x}) = p - p_c(\vec{x}) \leq 0,$$

Table 18 The comparison results for welded beam design problem

Algorithm	Optimum variables				Optimum cost
	h	l	t	b	
HH-ALO-Tabu	**0.205734718637559**	**3.25302960294620**	**9.03662391200579**	**0.205729639799296**	**1.695244924753503**
ALO	0.204222203731665	3.28333328856715	9.03693612658649	0.205758143324685	1.696925321278552
MVO	0.200768248690566	3.35175631600231	9.03895830751539	0.205798392227733	1.700393379516647
GWO	0.154333086549704	4.48115935216248	9.03665437392092	0.205729492624404	1.695668454126744
Chaotic-ALO	0.150495274625023	4.60652710565278	9.03662275626816	0.205729705442881	1.695313681433471

The bold refers to the model(s) with optimal cost

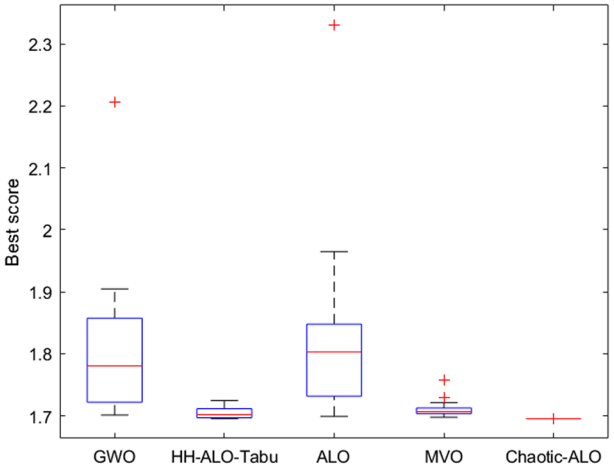

Fig. 55 Box plot for welded beam design problem

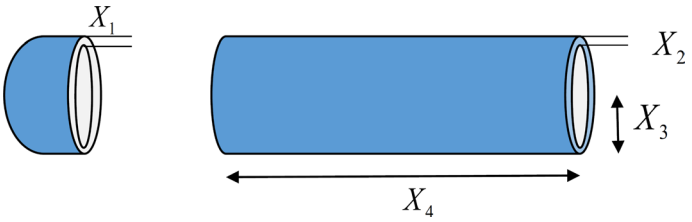

Fig. 56 Model of the pressure vessel (Abualigah et al. 2021b)

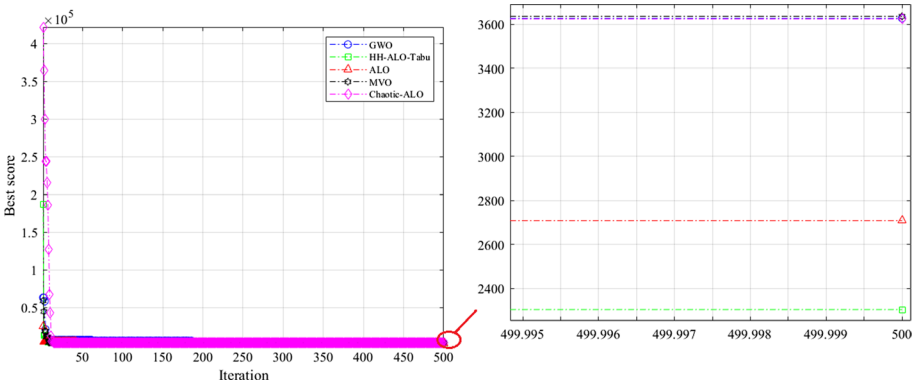

Fig. 57 Convergence curve of the compared methods on the pressure vessel problem

$$g_6(\vec{x}) = 0.125 - x_1 \leq 0,$$

Table 19 The comparison results for pressure vessel problem

Algorithm	Optimum variables				Optimum cost
	X_1	X_2	X_3	X_4	
HH-ALO-Tabu	**1.093813**	**0**	**65.225233**	**10**	**2.3025E+03**
ALO	0.461131	0	40.319618	200	2.6243E+03
MVO	0.489419	0	41.200853	188.25253	3.6361E+03
GWO	0.998408	0	61.322404	27.939410	3.7083E+03
Chaotic-ALO	0.467284	0	40.539089	196.9672	3.6271E+03

The bold refers to the model(s) with optimal cost

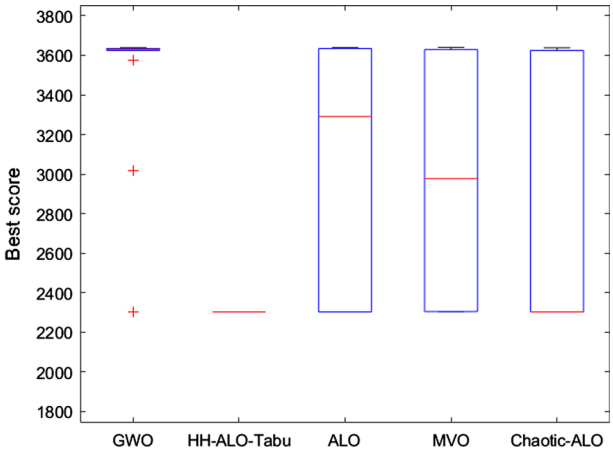

Fig. 58 Box plot for pressure vessel design problem

$$g_7(\vec{x}) = 1.10471x_1^2 = 0.04811x_3x_4(14.0 + x_2) \leq 0,$$

Variable range: $0.1 \leq x_1 \leq 2$

$$0.1 \leq x_2 \leq 10,$$

$$0.1 \leq x_3 \leq 10,$$

$$0.1 \leq x_4 \leq 2.$$

A welded beam design requires optimization of four variables, while the three-bar truss design requires optimization of only two variables. A convergence curve can be seen in Fig. 54 for different methods of solving a welded beam problem. Algorithms with step-like shapes escape from the local optimal. The last iteration showed that MVO, ALO, and Chaotic-ALO converged. A GWO algorithm achieves the worst result, while the HH-ALO-Tabu algorithm achieves the best result.

There is a presentation of the optimal value and cost for the different methods in Table 18. This method achieves a better performance than other methods and provides a

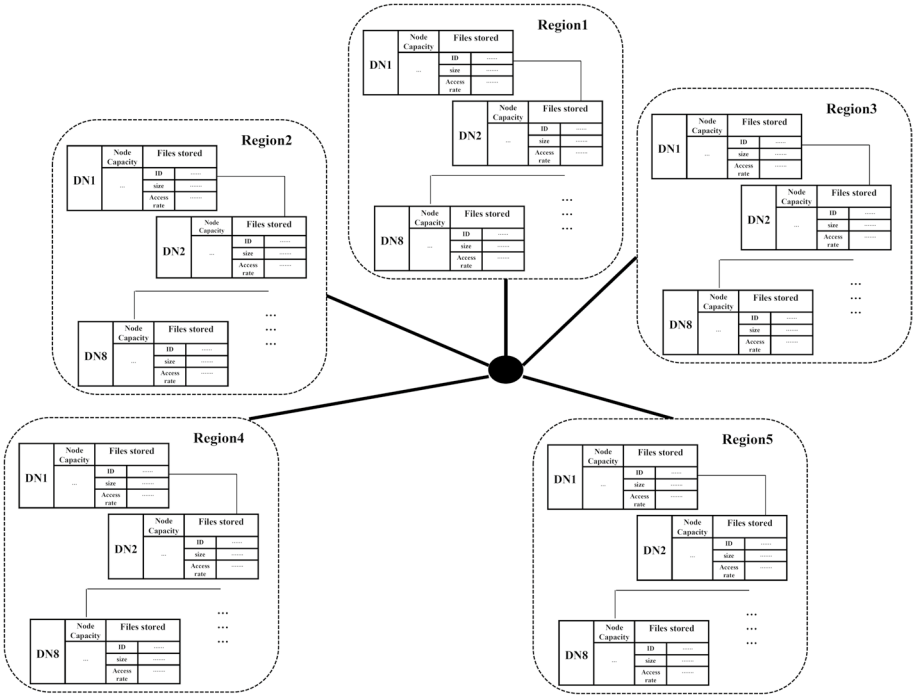

Fig. 59 Hierarchical network structure

better final result than MVO and ALO by 0.3% and 0.01%, respectively. A competitive result is achieved in comparison to GWO and Chaotic-ALO.

The box plot related to welded beam design for different methods is shown in Fig. 55. Chaotic-ALO and HH-ALO-Tabu show more reliability than other methods because their narrow form is more observable. The interquartile range and median of the proposed HH-ALO-Tabu and Chaotic-ALO are also superior to those of the others.

From Table 18 and Figs. 54 and 55, we can observe that the HH-ALO-Tabu is more stable, which indicates better robustness. An algorithm's robustness can be measured by its mean and STD. Due to the lower mean value of HH-ALO-Tabu than ALO, ALO's robustness is greater. The Chaotic-ALO calculates optimal values similarly to the HH-ALO-Tabu, but with a higher convergence rate than the HH-ALO-Tabu.

Making pressure vessels lighter is a design problem with the purpose of maximizing savings and ensuring safety. Figure 56 shows the structure and variables of the pressure vessel problem (Abualigah et al. 2021b).

$$\text{Consider: } [X_1\, X_2\ X_3\, X_4]$$

$$\text{Minimize: } f(x) = 0.6224X_1X_3X_4 + 1.7781X_2X_3^2 + 3.1661X_1^2X_4 + 19.84X_1^2X_3$$

$$\text{Subject to: } g_1(x) = 0.0193X_3 - X_1 \leq 0$$

Table 20 Default experiment parameters

Parameter	Value
Regions	5
Length of jobs	[1000–20,000]
Number of jobs processed	1000
Data nodes per region	8
Failure probability of data nodes	[0.015–0.5]
Data node processing capability	[1000–2000] MIPS
Intra-region bandwidth (Gbit/s)	[1.25–1.75]
Inter-region bandwidth (Gbit/s)	[0.25–0.5]
Jobs arrival delay	8 ms
Transfer rate of data nodes (MB/s)	[150–350]
Data node storage capacity	Parejo et al. (2012), Mirjalili et al. (2014), Kaveh et al. (2013), Kaveh and Khayatazad (2013), Kaveh and Mahdavi (2014), and Kaveh et al. (2014) GB
Intra-region transfer cost	$0.025/GB
Inter-region transfer cost	$0.25/GB
Storage cost	$0.20 to $0.30/GB
Processing cost	$1.25 to $2.25/$10^9$ MI
Number of files	[50–300]
Size of file	[1000–5000]
Size of Ant and Ant lion population	30
Search domain in the continuous version	[−1, 1]
Number of generations	250

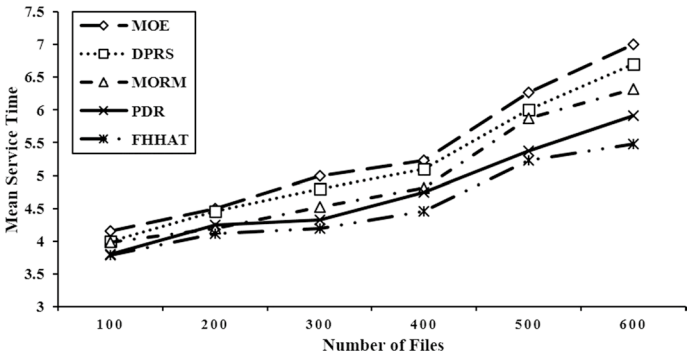

Fig. 60 Mean service time with a different number of files

$$g_2(x) = 0.00954X_3 - X_2 \leq 0,$$

$$g_3(x) = 1,296,000 - \pi X_3^2 X_4 - \frac{4}{3}\pi X_3^3 \leq 0,$$

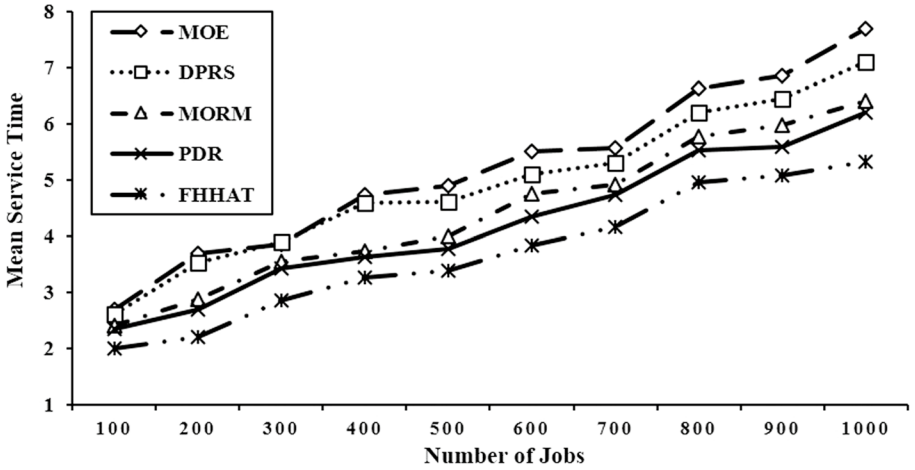

Fig. 61 Mean service time with a different number of jobs

$$g_4(x) = X_4 - 240 \leq 0,$$

$$\text{Variable range: } 0 \leq X_1 \leq 99, \quad 0 \leq X_2 \leq 99, \quad 10 \leq X_3, \quad X_4 \leq 200,$$

where X_1 and X_2 are uniformly discrete variables to show head (*Th*) and cylinder wall thickness (*Ts*), X_3 is a continuous variable to indicate the radius of the cylinder and head (R), X_4 shows the cylinder length (L), and is a continuous variable.

HH-ALO-Tabu is the only algorithm that finds the optimal solution to the pressure vessel problem without failure (Fig. 57). HH-ALO-Tabu cannot start from the appropriate location using the OBL strategy, but in the following iteration, a chaotic function and using different spiral paths lead to the optimal solution.

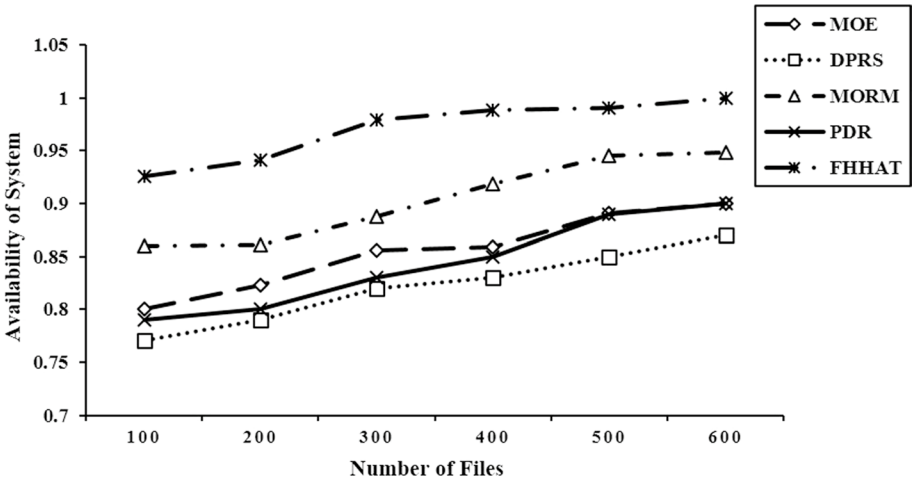

Fig. 62 Availability of system for different replication algorithms

Table 19 provides the optimum values for variables of pressure vessels obtained by different methods along with their final costs. It is easier to recognize the superiority of HH-ALO-Tabu as compared to the previous problem. As a result of the proposed HH-ALO-Tabu, the final results improve by 12.2%, 36.67%, 36.6%, and 37.90%, respectively, over ALO, MVO, Chaotic-ALO, and GWO.

A box plot illustrating the pressure vessel problem for different methods is shown in Fig. 58. As can be seen in Fig. 58, the best score of the proposed HH-ALO-Tabu is stable, since the interquartile range is narrow. Further, the 25th and 75th percentiles of the proposed algorithm samples declined toward the minimum solution during 30 runs. The convergence analysis (i.e., Fig. 57) indicates that HH-ALO-Tabu has promising convergence behavior. Therefore, HH-ALO-Tabu is the best among other methods for pressure vessel problem that is relatively difficult to optimize.

6.3 Experiment structure for evaluation of FHHAT algorithm

Based on the bandwidth hierarchy on the Internet, the proposed structure is derived. We assume the architecture for our experiments is similar to that presented in Fig. 59. It consists of eight datacenters geographically distributed in different regions. Compared with network links between regions, DNs within a region are located closer together and have a higher bandwidth available between them.

In order to simulate the FHHAT algorithm's performance, the CloudSim toolkit is used. Table 20 indicates the detailed setting of parameters in the simulation.

6.4 Simulation results

A comparison of the performance of MORM (Long et al. 2014), MOE (Hassan et al. 2009), DPRS (Mansouri et al. 2017), PDR (Mansouri and Javidi xxxx), and the proposed FHHAT algorithm was carried out in this section.

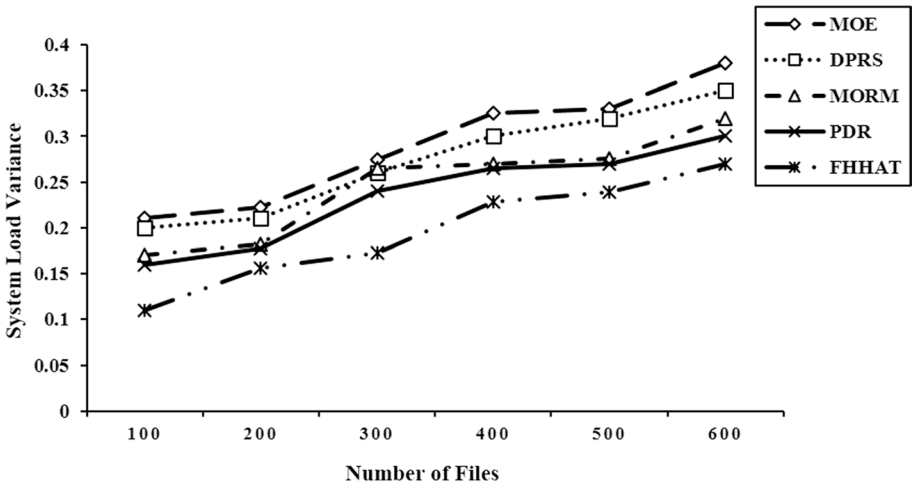

Fig. 63 Load variance for different replication algorithms

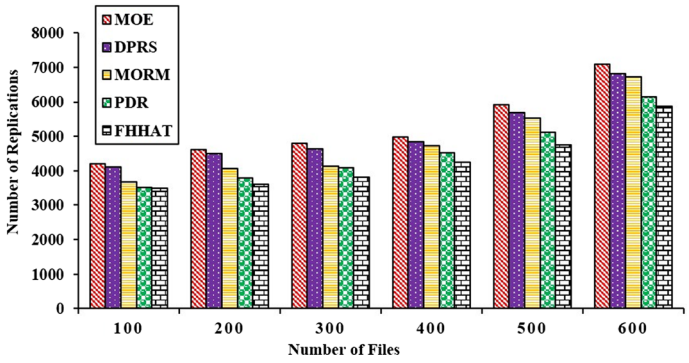

Fig. 64 Number of replications for different replication algorithms

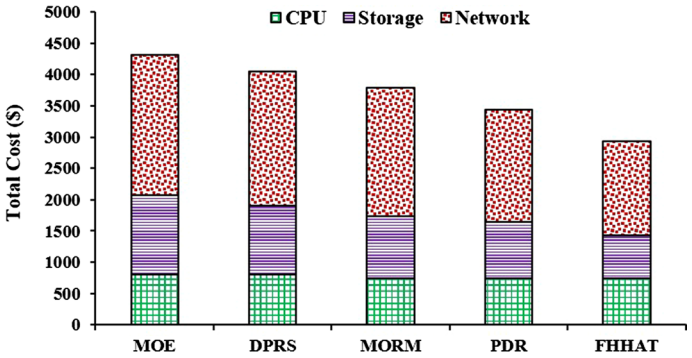

Fig. 65 Total cost of replication algorithms

6.4.1 Mean service time

We can conclude from Fig. 60 that replication strategies increase in service time as the number of files increases. We can see that the mean service time of the FHHAT algorithm is faster than other replication methods. In the case of 100 files, MOE takes 7 s, while FHHAT takes 5.5 s. When the number of files is 600 the response time of FHHAT is about 5.5 s, and the response time of MOE is about 7 s. As a result, FHHAT can deal with large files more efficiently than other algorithms, especially when the number of files is high, as in the cloud. According to critical parameters such as load, service time, and availability, the FHHAT algorithm determines the suitable number of replicas and placement of replicas.

Figure 61 shows the mean service time for various numbers of jobs. It shows that in a small number of jobs, FHHAT improves upon MOE by 24% while in a large number of jobs, it is a 31% improvement. In other words, with job numbers rising, the FHHAT strategy describes more advantages. This is because the replicas have already been stored on some nodes to balance out the load on the system.

6.4.2 System availability

The probability of a system being available is depicted in Fig. 62. Generally, as the number of replicas increases, the data availability also increases but the corresponding cost of replication increases. Compared with the DPRS method, the MORM strategy algorithm can increase the SA by around 8% on average, and increase around 5% on average than the MOE algorithm. Since the MORM strategy considers file availability as the main objective during the replication process. In a heavier workload, the FHHAT strategy achieves the best results in terms of SA. Because FHHAT solves the multi-objective data replication problem based on hyper-heuristics, it provides a good balance between several objectives (for example, SA, service time, load, and profit).

6.4.3 Load variance

According to Fig. 63, the proposed method has a better outcome than the other four methods based on the load variance. The main reason for FHHAT to perform better than other strategies is the way it takes into account load and network congestions. It is observed from Fig. 63 that when increasing the number of files, the advantages of the FHHAT algorithm are shown even more clearly. In this experiment, the load value of PDR and MORM are close to each other. While the DPRS shows about 7% less load variance than the MOE on average. As a result, the DPRS places the replicas according to the number of requests and the amount of storage space available.

6.4.4 Total number of replications

Figure 64 shows a comparison of the number of replications. It demonstrated that the presented algorithm reduces unnecessary replications when compared to other strategies. Since the placement of the replicas is done through an Ant Lion Optimization algorithm, the unnecessary replicas are getting reduced and the suitable number of replicas is maintained for each data file to get the near-optimal objective value. In addition, the FHHAT strategy replaces unpopular replicas from the regional point of view. In other words, when the selected site doesn't have enough storage space for storing a new replica and the file is duplicated in other sites within the region then it avoids replication. In contrast, DPRS and MOE both achieved reductions in the number of replications between 13 and 9%. Due to pre-replication, it is usually possible to execute a job on the local machine if the appropriate files are available.

6.4.5 Monetary cost

According to Fig. 65, the proposed replication algorithm performs well based on profit value. The good system has a maximum profit or minimum cost value. Since the proposed methodology considers the costs that are directly related to data replication (e.g., CPU, Storage, and Network costs), we can achieve a minimum cost of 2935$, which is very low compared to other algorithms. So, the cost is reduced by 30%, 27%, 22%, and 14% when compared to MOE, DPRS, MORM, and PDR strategies. The main reason is that the FHHAT algorithm tries to meet the performance and return a profit at the same time. Furthermore, it aims to reduce data transfer via interregional links by taking into account the differences between intra-regional and inter-regional bandwidths.

It is evident that data replication algorithms should provide an optimization model that serves both cloud consumers and cloud providers (e.g., service time). Due to its consideration of a scalar product between the weight vector and the objectives, the DPRS algorithm cannot cope with peak loads. Meta-heuristics are a powerful technique to address complex optimization problems. For example, it can be seen clearly that MORM significantly outperforms DPRS since it uses an improved artificial immune algorithm to solve multi-objective replication algorithms. But despite the good results of meta-heuristics, some of these algorithms suffer from low convergence rates and trapping in local optima. For example, the MOE algorithm uses NSGA-II for replica management. Nevertheless, the performance of the MOE algorithm is the worst in terms of replication metrics (e.g., service time). Another observation from the experiment is that the proposed replication algorithm has low monetary costs while satisfying user requirements. As a result of the fitness function of FHHAT, the algorithm has high storage, processing, and network cost.

FHHAT outperforms all other algorithms compared in the evaluations under different scales. This is because the proposed algorithm uses an efficient HH-ALO-Tabu approach based on important objectives (i.e., SA, service time, load variance, and PEs) for replica placement and applies the intelligence replica replacement process. In Sect. 6, the simulation results show that standard ALO has some limitations that degrade the performance of the algorithm. Therefore, a hybrid Ant Lion Optimizer combining Tabu search (ALO-Tabu) is proposed to achieve a balance between exploration and exploitation. ALO-Tabu presents better results compared with SSA, MVO, GWO, and WOA in terms of search history, convergence, trajectory, and average fitness of all agents. Since the proposed HH-ALO-Tabu creates the initial population based on CMs and OBL strategies. Moreover, it automatically selects a CM, OBL, and random walk strategy based on the DE algorithm. By balancing diversification and intensification, HH-ALO-Tabu achieves success. Further, by balancing exploration and exploitation abilities and robust search efficiency, the HH-ALO-Tabu proved applicable to real-life problems with satisfactory optimization results.

7 Conclusion

The main goal of the study is divided into two parts. Firstly, the standard ALO should be improved to increase its exploitation and exploration. A second challenge is how to place replicas in a cloud environment. The problems of standard ALO (i.e., premature convergence and stuck in local optimal) are related to the process of updating ants and ant lions wherein standard ALO each ant updates its location based on its random walk around a selected ant lion, and the best ant lion. Due to its dependency on the initial population, the ALO algorithm does not generate new solutions at the end of each iteration.

To solve these drawbacks, HH-ALO-Tabu is proposed by modifying the updating equation of ant lions so that the exploration activity of solutions is increased and ants are updated with several spiral paths which the order of choosing them is determined with a Tabu list. An appropriate chaotic function and an appropriate OBL strategy are used for the initialization of the first population. Deferential evolution is used to find the appropriate chaotic and OBL function.

The proposed HH-ALO-Tabu is evaluated with two types of measures (i.e., qualitative and quantitative) and two benchmark functions (i.e., CEC 2005 and CEC 2019) including several unimodal and multimodal functions. A three-bar truss, a welded beam design, and

a pressure vessel problem are used to demonstrate the capability of the proposed HH-ALO-Tabu in solving real problems.

The results from CEC 2005 benchmark functions indicate the HH-ALO-Tabu has a better performance compared to some variants of standard ALO (i.e., OBL-ALO, Chaotic-ALO, and GA-ALO) and can explore the search space better than these variants. The experiments from CEC 2019 show that HH-ALO-Tabu can improve the final results compared to some start-of-the-art algorithms (i.e., WOA, SSA, MVO, and GWO) and solve the premature convergence of standard ALO. The experiments based on three real problems (i.e., three-bar truss, the welded beam design, and pressure vessel problem) demonstrate that the proposed algorithm (HH-ALO-Tabu) is also applicable to the challenging problem with unknown search spaces. According to the results of the semi-real and real problems, the following conclusions can also be drawn:

- HH-ALO-Tabu can be suitable not only for unconstrained problems but also for constrained challenges.
- HH-ALO-Tabu can improve the target throughout iteration and hence approximates the global optimum accurately.
- HH-ALO-Tabu can enhance the initial random population for real problems.

Finally, we propose a multi-objective optimized replication algorithm (FHHAT) based on the HH-ALO-Tabu and fuzzy system that determines a reasonable number of replicas and the optimal locations for them by considering service time, SA, load, and total costs of the provider. The majority of existing replication algorithms focus only on performance factors, such as response time, and ignore the cloud provider's economic profit. The FHHAT balances the benefits and costs of the replication. FHHAT replaces unpopular files from the regional perspective if adequate storage space isn't available on the selected site.

We evaluate the proposed replication strategy using the CloudSim toolkit and our results indicate that FHHAT has a better performance compared with MORM, MOE, PDR, and PDRS in terms of the number of replications, load variance, average service time, and cost. The main reason is that FHHAT stores new replicas in a location that ensures tenant performance expectations are met. Furthermore, it avoids wasting resources such as storage and retires unnecessary copies accordingly. For determining a reasonable number and location of replicas, we will consider a reliability model as well as an energy consumption model in the future. Consistency protocols can be further improved to provide high consistency among replicas of a data file.

References

Abd Elaziz M, Mirjalili S (2019) A hyper-heuristic for improving the initial population of whale optimization algorithm. Knowl Based Syst 172:42–63. https://doi.org/10.1016/j.knosys.2019.02.010

Abualigah L, Diabat A (2021) Advances in Sine Cosine Algorithm: a comprehensive survey. Artif Intell Rev 54:2567–2608. https://doi.org/10.1007/s10462-020-09909-3

Abualigah L, Dulaimi AJ (2021) A novel feature selection method for data mining tasks using hybrid Sine Cosine Algorithm and Genetic Algorithm. Clust Comput. https://doi.org/10.1007/s10586-021-03254-y

Abualigah L, Diabat A, Mirjalili S, Elaziz MA, Gandomi AH (2021a) The arithmetic optimization algorithm. Comput Methods Appl Mech Eng. https://doi.org/10.1016/j.cma.2020.113609

Abualigah L, Yousri D, Elaziz MA, Ewees AA, Al-qaness MAA, Gandomi AH (2021b) Aquila Optimizer: a novel meta-heuristic optimization algorithm. Comput Ind Eng. https://doi.org/10.1016/j.cie.2021.107250

Alami Milani B, Jafari Navimipour N (2016) A comprehensive review of the data replication techniques in the cloud environments: major trends and future directions. Netw Comput Appl 64:229–238. https://doi.org/10.1016/j.jnca.2016.02.005

Alatas B, Akin E, Ozer B (2009) Chaos embedded particle swarm optimization algorithms. Chaos Solitons Fractals 40(4):1715–1734. https://doi.org/10.1016/j.chaos.2007.09.063

Alcalá-Fdez J, Sánchez L, García S, del Jesus MJ, Ventura S, Garrell JM, Otero J, Romero C, Bacardit J, Rivas VM, Fernández JC, Herrera F (2009) KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput 13:307–318. https://doi.org/10.1007/s00500-008-0323-y

Arena P, Caponetto R, Fortuna L, Rizzo A, La Rosa M (2000) Self-organization in non-recurrent complex system. Int J Bifurc Chaos 10(5):1115–1125. https://doi.org/10.1142/S0218127400000785

Beigrezaei M, Haghighat AT, Mirtaheri SL (2021) Improve performance by a fuzzy-based dynamic replication algorithm in grid, cloud, and fog. Math Probl Eng. https://doi.org/10.1155/2021/5522026

Borthakur D (2007) The Hadoop distributed file system: architecture and design, Hadoop Project Website

Branco Jr T, de Sá-Soaresa F, Lopez Rivero A (2017) Key issues for the successful adoption of cloud computing. In: International conference on enterprise information systems, 2017, vol 121, pp 115–122. https://doi.org/10.1016/j.procs.2017.11.016

Chunlin L, Ping WY, Hengliang T, Youlong L (2019) Dynamic multi-objective optimized replica placement and migration strategies for SaaS applications in edge cloud. Future Gener Comput Syst 100:921–937. https://doi.org/10.1016/j.future.2019.05.003

Coelho LS, Mariani VC (2008) Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. Expert Syst Appl 34:1905–1913. https://doi.org/10.1016/j.eswa.2007.02.002

Coelho LS, Mariani VC (2012) Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. Comput Math Appl 64(8):2371–2382. https://doi.org/10.1016/j.camwa.2012.05.007

Dong H, Xu Y, Li X, Yang Z, Zou C (2021) An improved antlion optimizer with dynamic random walk and dynamic opposite learning. Knowl Based Syst. https://doi.org/10.1016/j.knosys.2021.106752

dos Santos CL, Mariani VC (2008) Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. Expert Syst Appl 34(3):1905–1913. https://doi.org/10.1016/j.eswa.2007.02.002

Du Z, Hu J, Chen Y, Cheng Z, Wang X (2011) Optimized QoS-aware replica placement heuristics and applications in astronomy data grid. J Syst Softw 84(7):1224–1232. https://doi.org/10.1016/j.jss.2011.02.038

Emary E, Zawbaa HM, Hassanien AE (2016) Binary ant lion approaches for feature selection. Neurocomputing 213:54–65. https://doi.org/10.1016/j.neucom.2016.03.101

Ergezer M, Simon D, Du D (2009) Oppositional biogeography-based optimization. In: IEEE international conference on systems, man and cybernetics, 2009. https://doi.org/10.1109/ICSMC.2009.5346043

Gandomi AH, Yang XS, Alavi AH (2013a) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29:17–35. https://doi.org/10.1007/s00366-011-0241-y

Gandomi AH, Yun GJ, Yang XS, Talatahari S (2013b) Chaos-enhanced accelerated particle swarm optimization. Commun Nonlinear Sci Numer Simul 18(2):327–340. https://doi.org/10.1016/j.cnsns.2012.07.017

García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Inf Sci 180:2044–2064. https://doi.org/10.1016/j.ins.2009.12.010

Gopinath S, Sherly E (2018) A dynamic replica factor calculator for weighted dynamic replication management in cloud storage systems. In: International conference on computational intelligence and data science, 2018, vol 132, pp 1771–1780. https://doi.org/10.1016/j.procs.2018.05.152

Goyal T, Singh A, Agrawal A (2012) CloudSim: simulator for cloud computing infrastructure and modeling. Procedia Eng 38:3566–3572. https://doi.org/10.1016/j.proeng.2012.06.412

Han P, Du C, Jinchao C, Ling F, Du X (2021) Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique. J Syst Archit. https://doi.org/10.1016/j.sysarc.2020.101837

Hassan OA-H, Ramaswamy L, Miller J, Rasheed K, Canfield ER (2009) Replication in overlay networks: a multi-objective optimization approach. In: Collaborative computing: networking, applications and worksharing, lecture notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2009, vol 10, pp 512–528. https://doi.org/10.1007/978-3-642-03354-4_39

Jain M, Singh V, Rani A (2019) A novel nature-inspired algorithm for optimization: squirrel search algorithm. Swarm Evol Comput 44:148–175. https://doi.org/10.1016/j.swevo.2018.02.013

Jayasree P, Saravanan V (2018) APSDRDO: adaptive particle swarm division and replication of data optimization for security in cloud computing. IOSR J Eng 2278–8719

Kacimi MA, Guenounou O, Brikh L, Yahiaoui F, Hadid N (2020) New mixed-coding PSO algorithm for a self-adaptive and automatic learning of Mamdani fuzzy rules. Eng Appl Artif Intell. https://doi.org/10.1016/j.engappai.2019.103417

Kaucic M (2013) A multi-start opposition-based particle swarm optimization algorithm with adaptive velocity for bound constrained global optimization. J Glob Optim 55(1):165–188. https://doi.org/10.1007/s10898-012-9913-4

Kaveh A (2014) Dolphin echolocation optimization. Adv Metaheuristic Algorithms Optim Des Struct 59:157–193. https://doi.org/10.1016/j.advengsoft.2013.03.004

Kaveh A, Khayatazad M (2013) Ray optimization for size and shape optimization of truss structures. Comput Struct 117:82–94. https://doi.org/10.1016/j.compstruc.2012.12.010

Kaveh A, Mahdavi V (2014) Colliding bodies optimization method for optimum design of truss structures with continuous variables. Adv Eng Softw 70:1–12. https://doi.org/10.1016/j.advengsoft.2014.01.002

Kaveh A, Zolghadr A (2014) Democratic PSO for truss layout and size optimization with frequency constraints. Comput Struct 130:10–21. https://doi.org/10.1016/j.compstruc.2013.09.002

Kaveh A, Motie M, Moslehi M (2013) Magnetic charged system search: a new meta-heuristic algorithm for optimization. Acta Mech. https://doi.org/10.1007/s00707-012-0745-6

Kaveh A, Bakhshpoori T, Afshari E (2014) An efficient hybrid particle swarm and swallow swarm optimization algorithm. Comput Struct 143:40–59. https://doi.org/10.1016/j.compstruc.2014.07.012

Khalili Azimi S (2019a) A bee colony (beehive) based approach for data replication in cloud environments. Fundam Res Electr Eng. https://doi.org/10.1007/978-981-10-8672-4_80

Khalili Azimi S (2019b) A bee colony (beehive) based approach for data replication in cloud environments. Fundam Res Electr Eng. https://doi.org/10.1007/978-981-10-8672-4_80

Kılıç H, Yüzgeç U (2019) Tournament selection based antlion optimization algorithm for solving quadratic assignment problem. Eng Sci Technol Int J 22(2):673–691. https://doi.org/10.1016/j.jestch.2018.11.013

Kwame Senyo P, Addae E, Boateng R (2018) Cloud computing research: a review of research themes, frameworks, methods and future research directions. Int J Inf Manag 38(1):128–139. https://doi.org/10.1016/j.ijinfomgt.2017.07.007

Liang J, Suganthan P, Deb K (2005) Novel composition test functions for numerical global optimization. In: Proceedings 2005 IEEE swarm intelligence symposium, 2005, pp 68–75. https://doi.org/10.1109/SIS.2005.1501604

Liu H, Cai Z, Wang Y (2010) Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 10:629–640. https://doi.org/10.1016/j.asoc.2009.08.031

Long SQ, Zhao YL, Chen W (2014) MORM: A Multi-objective Optimized Replication Management strategy for cloud storage cluster. J Syst Archit 60(2):234–244. https://doi.org/10.1016/j.sysarc.2013.11.012

Luo Y, Li R, Zhang L, Tian F (2004) Application of artificial immune algorithm to function optimization. In: Fifth world congress on intelligent control and automation, 2004. https://doi.org/10.1109/WCICA.2004.1341989

Mahdavi Jafari M, Khayati GR (2018) Prediction of hydroxyapatite crystallite size prepared by sol–gel route: gene expression programming approach. J Sol–Gel Sci Technol 86(1):112–125. https://doi.org/10.1007/s10971-018-4601-6

Manganaro G, de Gyvez JP (1997) DNA computing based on chaos. In: IEEE international conference on evolutionary computation, 1997, pp 255–260. https://doi.org/10.1109/ICEC.1997.592306

Mansouri N (2014) A threshold-based dynamic data replication and parallel job scheduling strategy to enhance data grid. Clust Comput 17(3):957–977. https://doi.org/10.1007/s10586-013-0330-3

Mansouri N (2016) Adaptive data replication strategy in cloud computing for performance improvement. Front Comput Sci 10(5):925–935. https://doi.org/10.1007/s11704-016-5182-6

Mansouri N, Javidi MM (2017) A new prefetching-aware data replication to decrease access latency in cloud environment. J Syst Softw 144:197–215. https://doi.org/10.1016/j.simpat.2017.06.001

Mansouri N, Javidi MM (2018) A hybrid data replication strategy with fuzzy-based deletion for heterogeneous cloud data centers. J Supercomput 74(10):5349–5372. https://doi.org/10.1007/s11227-018-2427-1

Mansouri N, Kuchaki Rafsanjani M, Javidi MM (2017) DPRS: a dynamic popularity aware replication strategy with parallel download scheme in cloud environments. Simul Model Pract Theory 77:177–196. https://doi.org/10.1016/j.simpat.2017.06.001

Mansouri N, Javidi MM, Mohammad Hasani Zade B (2020) Using data mining techniques to improve replica management in cloud environment. Soft Comput 24:7335–7360. https://doi.org/10.1007/s00500-019-04357-w

Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98. https://doi.org/10.1016/j.advengsoft.2016.01.008

Mirjalili S, Lewis A (2016) The Whale optimization algorithm. Adv Eng Softw 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27(2):495–513. https://doi.org/10.1007/s00521-015-1870-7

Mohammed A, Duffuaa SO (2020) A Tabu search based algorithm for the optimal design of multi-objective multi-product supply chain networks. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2019.07.025

Mohammad Hasani Zade B, Mansouri N, Javidi MM (2021) SAEA: a security-aware and energy-aware task scheduling strategy by Parallel Squirrel Search Algorithm in cloud environment. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2021.114915

Mousa AA, El-Shorbag MA, Farag MA (2020) Steady-state sine cosine genetic algorithm based chaotic search for nonlinear programming and engineering applications. IEEE Access 8:212036–212054. https://doi.org/10.1109/ACCESS.2020.3039882

Parejo A, Ruiz-Cortés A, Lozano S, Fernandez P (2012) Metaheuristic optimization frameworks: a survey and benchmarking. Soft Comput 16(3):527–561. https://doi.org/10.1007/s00500-011-0754-8

Phan DH, Suzuki J, Carroll R (2012) Evolutionary multiobjective optimization for green clouds. In: Proceedings of the 14th annual conference companion on genetic and evolutionary computation, 2012, pp 19–26. https://doi.org/10.1145/2330784.2330788

Pradhan R, Kumar Majhi S, Ku Pradhan J, Bhusan Pati B (2018) Antlion optimizer tuned PID controller based on Bode ideal transfer function for automobile cruise control system. J Ind Inf Integr 9:45–52. https://doi.org/10.1016/j.jii.2018.01.002

Price KV, Awad NH, Ali MZ, Suganthan PN (2018) The 100-digit challenge: problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization. Nanyang Technological University, Nanyang

Rahnamayan S, Tizhoosh HR, Salama MM (2007) Quasi-oppositional differential evolution. In: Proceedings of the IEEE congress on evolutionary computation, 2007, pp 2229–2236. https://doi.org/10.1109/CEC.2007.4424748

Ren B, Zhong W (2011) Multi-objective optimization using chaos based PSO. J Inf Technol 10(10):1908–1916. https://doi.org/10.3923/itj.2011.1908.1916

Sadollah A, Bahreininejad A, Eskandar H, Hamdi M (2013) Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 13:2592–2612. https://doi.org/10.1016/j.asoc.2012.11.026

Saha S, Mukherjee V (2018) A novel quasi-oppositional chaotic antlion optimizer for global optimization. Appl Intell 48(9):2628–2660. https://doi.org/10.1016/j.knosys.2021.106752

Saranya C, Manikandan G (2013) A study on normalization techniques for privacy preserving data mining. Int J Eng Technol 5:2701–2704

Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimization algorithm: theory and application. Adv Eng Softw 105:30–47. https://doi.org/10.1016/j.advengsoft.2017.01.004

Seif Z, Ahmadi M (2015) An opposition-based algorithm for function optimization. Eng Appl Artif Intell 37:293–306. https://doi.org/10.1016/j.engappai.2014.09.009

Sousa FRC, Machado JC (2012) Towards elastic multi-tenant database replication with quality of service. In: IEEE/ACM 5th international conference on utility and cloud computing, 2012, pp 168–175. https://doi.org/10.1109/UCC.2012.36

Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. Nat Comput. https://doi.org/10.1007/s11047-018-9704-z

Suneel M (2006) Chaotic sequences for secure CDMA. Ramanujan Institute for Advanced Study in Mathematics, Chennai, pp 1–4

Tang M, Sung Lee B, Kiat Yeo C, Tang X (2005) Dynamic replication algorithms for the multi-tier data grid. Future Gener Comput Syst 21(5):775–790. https://doi.org/10.1016/j.future.2004.08.001

Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence. In: International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, Web technologies and Internet commerce (CIMCA-IAWTIC'06), 2005. https://doi.org/10.1109/CIMCA.2005.1631345

Tos U, Mokadem R, Hameurlain A, Ayav T, Bora S (2018) Ensuring performance and provider profit through data replication in cloud systems. Clust Comput 21(3):479–1492. https://doi.org/10.1007/s10586-017-1507-y

Vulimiri A, Curino C, Godfrey B, Jungblut T, Padhye J, Varghese G (2015) Global analytics in the face of bandwidth and regulatory constraints. In: 12th USENIX symposium on networked systems design and implementation, 2015, pp 323–336

Wang T, Yao S, Xu Z, Pan S (2017) Dynamic replication to reduce access latency based on fuzzy logic system. Comput Electr Eng 60:48–57. https://doi.org/10.1016/j.compeleceng.2016.11.022

Wang M, Heidari AA, Chen M, Chen H, Zhao X, Cai X (2020) Exploratory differential ant lion-based optimization. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2020.113548

Wei Sun D, Chang GR, Gao S, Jin LZ, Wei Wang X (2012) Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. J Comput Sci Technol 27(2):256–272. https://doi.org/10.1007/s11390-012-1221-4

Wu X (2017) Combination replicas placements strategy for data sets from cost-effective view in the cloud. Int J Comput Intell Syst 10:521–539. https://doi.org/10.2991/ijcis.2017.10.1.36

Xie F, Yan J, Shen J (2017) Towards cost reduction in cloud-based workflow management through data replication. In: Fifth international conference on advanced cloud and big data (CBD), 2017. https://doi.org/10.1109/CBD.2017.24

Yao S, Wen S, Yao B, Li XY (2018) DARS: a dynamic adaptive replica strategy under high load cloud-P2P. Future Gener Comput Syst 78:31–40. https://doi.org/10.1016/j.future.2017.07.046

Zawbaa HM, Emary E, Grosan C (2016) Feature selection via chaotic antlion optimization. PLoS ONE. https://doi.org/10.1371/journal.pone.0150652

Zhang M, Luo W, Wang X (2008) Differential evolution with dynamic stochastic selection for constrained optimization. Inf Sci 178:3043–3074. https://doi.org/10.1016/j.ins.2008.02.014