Evolutionary Intelligence (2021) 14:1997–2025
https://doi.org/10.1007/s12065-020-00479-5

RESEARCH PAPER

Improved chaotic binary grey wolf optimization algorithm for workflow scheduling in green cloud computing

Ali Mohammadzadeh[1] · Mohammad Masdari[1] · Farhad Soleimanian Gharehchopogh[1] · Ahmad Jafarian[2]

Received: 3 April 2020 / Revised: 9 July 2020 / Accepted: 25 August 2020 / Published online: 11 September 2020

Abstract
The workflow scheduling in the cloud computing environment is a well-known NP-complete problem, and metaheuristic algorithms are successfully adapted to solve this problem more efficiently. Grey wolf optimization (GWO) is a recently proposed interesting metaheuristic algorithm to deal with continuous optimization problems. In this paper, we proposed IGWO, an improved version of the GWO algorithm which uses the hill-climbing method and chaos theory to achieve better results. The proposed algorithm can increase the convergence speed of the GWO and prevents falling into the local optimum. Afterward, a binary version of the proposed IGWO algorithm, using various S functions and V functions, is introduced to deal with the workflow scheduling problem in cloud computing data centers, aiming to minimize their executions' cost, makespan, and the power consumption. The proposed workflow scheduling scheme is simulated using the CloudSim simulator and the results show that our scheme can outperform other scheduling approaches in terms of metrics such as power consumption, cost, and makespan.

Keywords Meta-heuristic · Grey wolf optimization · Green cloud computing · Workflow scheduling

1 Introduction

Cloud computing provides an interesting technology that makes scientific and industrial projects easier to implement. Infrastructure as a service (IaaS) is a type of cloud computing that provides online resources for virtualized computing. In addition to software as a service (SaaS) and platform as a service (PaaS), IaaS is one of the three main categories of cloud computing services [1]. Cloud computing can be used to deploy highly complex applications for scientific workflow. Workflows break down complex, data-intensive applications into smaller tasks and perform them in serial or parallel depending on the application's nature. Workflow models are commonly used in fields such as science, business, and engineering. In the planning of the scientific workflow, we need to take the following questions: (1) how to allocate tasks to VMs; (2) in what order the VMs will perform tasks taking into account the data dependency between tasks. Usually, these scientific workflows require a great deal of data of different sizes and simulations of long-term computers. We need high computing power and the availability of large infrastructure that offers different levels of QoS for the grid and more recently cloud computing environments.

There are various strategies to solving the problem of scheduling; scheduling schemes can usually be defined as follows: metaheuristic-based scheduling, heuristic workflow scheduling, hybrid metaheuristic, and heuristic scheduling, and Task and workflow scheduling [2]. Metaheuristic scheduling schemes can use algorithms such as simulated annealing (SA), ant colony optimization (ACO), and particle swarm optimization (PSO) to generate optimum schedules. For metaheuristic scheduling workflow schemes, the goals sought are often found in the metaheuristic algorithm's fitness function. For this reason, once multiple objections are implemented to scheduling, different coefficients are

✉ Mohammad Masdari
M.Masdari@iaurmia.ac.ir

Ali Mohammadzadeh
A.mohammadzadeh@iaurmia.ac.ir

Farhad Soleimanian Gharehchopogh
Bonab.Farhad@gmail.com

Ahmad Jafarian
A.Jafarian@iaurmia.ac.ir

[1] Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran

[2] Department of Mathematics, Urmia Branch, Islamic Azad University, Urmia, Iran

allocated for each goal that can change the effect of each goal on the overall scheduling of workflow. Also, Metaheuristic algorithms were commonly used to solve various optimization issues, such as the selection of features [3].

The metaheuristic optimization algorithms have become so popular over the past two decades. Some of these algorithms are even known among scholars from other sciences. They are usually inspired by physical phenomena and animals' behaviors. Additionally, learning, using, and combining these algorithms are done easily. The only point in using these algorithms is how to give input and get output from the system. These algorithms act better than conventional optimization methods in the comparison with local optimizations given their stochastic nature. On the contrary, all meta-heuristic methods have some weaknesses [4–7].

A common feature among the meta-heuristic approaches is dividing the search process into two stages: exploration and exploitation [8]. The exploration step is the broad random search in the search space to reach more desirable segments. The exploitation step is the local and more precise search in the desirable explored segments of the search space. Finding the right balance between these two stages is one of the challenging issues of metaheuristic methods.

Several approaches have been proposed concerning the metaheuristic algorithms in the past few decades, which we intend to review in some cases. Particle swarm optimization (PSO) algorithm [9, 10] is inspired by the collective behavior of the birds where each particle tries to find the best response in the search space by changing its speed and direction. Gandomi [11] introduced the krill herd (KH) optimization algorithm according to krill feeding behavior. In this algorithm, particle motion is specified according to three factors: food search behavior, random scattering, and the motion created by other particles [11]. Bat algorithm is inspired by the echolocation behavior of bats that was proposed by Yang in 2012. In this algorithm, the particles randomly navigate the search space at a different speed, position, frequency, and sound band [12].

Ant lion optimizer (ALO) algorithm [13] was first proposed by mirjalili according to ant lion feeding behavior for continuous problems. Ant lion performance uses specific movements of this insect to trap the ants in a pit. Multi-verse optimizer (MVO) algorithm is a kind of the new powerful meta-heuristic algorithms inspired by nature, where it has been commonly applied in a lot of fields, and based on three cosmological concepts: black hole, a white hole, and wormhole. These three concepts are used for exploration, exploitation, and local search [14]. Moth-flame optimizer algorithm [15] is a population-based algorithm that is proposed by Mirjalili. The moths in this algorithm move in a single, two, or multidimensional space. Cosine and logarithmic spiral function are used to implement moth-flame motion.

In the shuffled frog leaping algorithm [16], the frogs mimic each other and try to improve this behavior locally and turn into a model that others can imitate. This is a conscious imitation and not a mere imitation. The new optimization algorithm, inspired by the static and dynamic behavior of dragonflies' accumulation and its two main phases of exploration and exploitation, has been done by modeling the grasshopper optimization algorithm's social relationship in guiding, searching for food, and avoiding the enemies [17].

Several papers have been presented regarding grey wolves in recent years. In [18], a hybrid algorithm of PSO and GWO algorithms is presented. The particle swarm position is first updated by the PSO algorithm and then by the GWO algorithm. In [19], Kumar et al. proposed three strategies for improving the GWO algorithm. The first strategy is to use weighted baits. He then uses the laws of astrophysics to better guide the grey wolves to more desirable goals. According to this strategy, the wolves perform both exploration and exploitation processes. In the third strategy, the features and benefits of the two strategies are applied together. Various schemes have also been proposed to improve the GWO algorithm [20, 21].

The basic GWO algorithm does not perform well in the identification and exploration of global optimums that affect the convergence rate, despite good convergence. Hence, the purpose of the paper is to present the improved GWO algorithm that performs better in global optimizations. This algorithm is based on collective intelligence and is inspired by the collective behavior and hunting of grey wolves in nature. The innovation of the paper is using hill-climbing problem and random numbers based on chaos problem in the proposed algorithm. With the improvements done, we have seen good results with the basic method based on the chaos theory of the GWO algorithm. Many papers have used random numbers based on chaos problem and hill-climbing to improve the optimization algorithms [22–28].

The main contributions of this paper are as follows: (1) inspired by the GWO algorithm, an improved version of this algorithm was designed in this paper. The proposed algorithm is search improvement using hill-climbing problem and random numbers based on the theory of chaos. In this scheme, the update of the particle's position at each iteration is affected by the last few positions. (2) We used some standard mathematical optimization benchmark functions to compare the other algorithms. These benchmark functions were chosen as single exponential, multi-exponential, and finite-dimensional, with varying levels of hardness. (3) Using the transfer functions in the proposed algorithm for the solution of scientific workflow scheduling. (4) To evaluate the performance of the proposed algorithm practically, we implemented the extended proposed algorithm using the WorkflowSim tool, which is based on the CloudSim tool.

The results of the proposed algorithm were compared with some other algorithms.

The rest of this paper is organized as follows. The second section explores the context and the related work in recent literature. This section divided into two parts: heuristic, and meta-heuristic algorithms. The third section briefly explains the basic GWO algorithm. The proposed algorithm is provided in the fourth part, and in this part, we describe the innovation in the proposed algorithm, such as the improvement in search using hill-climbing problem and random numbers based on chaos theory. In the fifth part, the optimization functions and simulation results will be discussed. Section 5 explains the simulation component in detail, focusing on the optimization functions which divided into three groups: unimodal, multimodal, and constrained multimodal benchmark functions. Part six is a workflow scheduling in green cloud computing. In this section, we highlight the scheduling problem and describe a binary version of the proposed algorithm to use in discrete problems. In this section, we integrate the S-shaped and V-shaped transfer functions into the algorithm to convert the continuous proposed algorithm into the binary version, and simulation of workflow in the cloud-sim simulator are presented in this section, respectively. Section seven and eight exhibits the discussing and concluding remarks.

2 Literature review

Workflow scheduling is a NP-complete problem. Various heuristic and meta-heuristic algorithms have been proposed that solve workflow scheduling problems. The rest of this section is classified as heuristic and meta-heuristic algorithms.

2.1 Heuristic algorithms

Many heuristic algorithms such as MIN–MIN [29], and MAX–MIN [30] are proposed in the scheduling literature. List scheduling is one of the most common methods of scheduling workflow [31], in which a priority is given to each of the workflow tasks. Among the list-based heuristic algorithms, the critical path on the processor (CPOP) [32], dynamic level scheduling (DLS) [33], heterogeneous earliest finish time (HEFT) [34], dynamic heterogeneous earliest finish time (DHEFT) [35] and dynamic critical path (DCP) can be cited [36]. All these algorithms are aimed to reduce workflow makespan [37]. Some papers have considered two main criteria such as makespan and cost for scheduling. In [38], a multi-objective list-scheduling algorithm is presented to find dominant responses using Pareto for heterogeneous environments.

There are many bi-objective and multi-objective heuristic algorithms which solve workflow scheduling problem. The following are some examples of these algorithms. A bi-objective dynamic level scheduling algorithm performs the assignment of tasks in conditions where the runtime is acceptable against reliability [39]. The authors considered makespan and cost and used the concept of Pareto dominance to run large cloud applications to reduce financial costs regardless of budget and time constraints.

Camelo et al. [40], have presented a multi-objective heuristic algorithm to solve the workflow scheduling problem in the grid environment and they have used branch and bound's deterministic algorithm to find real Pareto front solutions. Juan et al. [37], have enhanced a new multi-objective list-based scheduling algorithm to deal with numerous contradictory objectives such as makespan, cost, and suggested a genuinely multi-criteria optimization reaching the Pareto front using a variety of well-distributed trading solutions chosen from the crowding distance and analyzed that use the HV metric. Multi-objective HEFT [41] has been provided for scheduling workflows on Amazon EC2. Cost and makespan have been considered to create an optimal Pareto and the users have been allowed to select the best solution manually.

2.2 Meta-heuristic algorithms

Matthews et al. [42], have proposed a new scheduling algorithm based on ant colony algorithm, aimed to minimize the workflow time and the makespan. Unfortunately, these have been ignored job priorities. In [43], a cost-based algorithm is presented for efficient mapping of tasks to available cloud resources. This scheduling algorithm considers the cost of resource use and efficiency. The scheduler divides all tasks by priority into three categories: high, medium, and low priorities.

Mozmar et al. [44], have proposed a bi-objective hybrid genetic scheduling algorithm for parallel applications, limited priority in heterogeneous distributed systems like cloud computing infrastructure. One of the advantages of this algorithm is the reduction of makespan and communication costs.

One way to solve multi-objective problems is to convert them to weighted single-objective problems. In [45], Li et al. transform the multi-objective problem into a single-objective problem with the heuristic algorithm. They have been focused on using cloud resources to provide large-scale graph computing tasks. This algorithm produces a priority task list and passes the highest priority task in a cloud environment to a cost-effective virtual machine.

Dangra et al. [46], have proposed a scheduling method with the technique of transforming a multi-objective problem into single-objective to increase efficiency and reliability. They have proposed a reliable dynamic level scheduling

(RDLS) algorithm based on dynamic level scheduling (DLS) [47]. Unlike the previous method where only one definite solution is proposed as a result of the algorithm, a set of dominant solutions provided to the user.

Yu et al. [48], have used the multi-objective evolutionary algorithm (MOEA) to solve the workflow scheduling problem. This algorithm is used to reduce two contradictory criteria of cost and runtime. Besides these two criteria, budget constraints and deadlines are considered in the algorithm as well. Population-based algorithms SPEA2 and NSGA-II [49, 50] and local search algorithms such as MOEA and PAES [51] have been used to solve workflow scheduling problems with conflicting objectives and various constraints. Another multi-objective algorithm with the R-NSGA-II approach [52] obtains Pareto optimal solutions according to three conflicting objectives: runtime, total cost, and reliability.

By examining the past studies, one can conclude that most multi-objective heuristic algorithms are suitable for the grid model and studies on the cloud environment are few. On the other hand, most studies have been conducted to reduce makespan and cost, ignoring the energy issue. In their proposed approach, Khalili et al. [53], have considered throughput besides these two criteria. In our proposed method, besides reducing the makespan and implementation cost, the consumed energy and throughput have been considered as criteria. This paper presents a multi-objective optimization solution to create optimal Pareto responses for the workflow in the green cloud environment.

3 Grey wolf optimization algorithm

This algorithm imitates the leadership hierarchy and grey wolves' hunting mechanism in nature. In this algorithm, four types of a grey wolf—alpha, beta, delta, and omega—are used to simulate the leadership hierarchy. Moreover, three main stages of hunting—Hunt for a target, encircling target, and attacking target—are simulated. According to Moro et al., the main stages of grey wolves' hunting are as follows [54]: tracking, chasing, and approaching prey. Chasing, seizing, and teasing the prey until it stops moving and attacking the prey. For mathematical modeling of the social hierarchies of wolves, we name the most appropriate solution as alpha, and among the best solutions, we name the second and third as beta and delta, respectively. The rest of the candidate solutions are considered as omega. The optimization process is directed by alpha, beta, and delta, and the fourth group follows these three groups. Equations 1 and 2 are used to model wolf siege behavior [54]:

$$\vec{D} = \left|\vec{C}.\vec{X}_p(t) - \vec{X}(t)\right| \tag{1}$$

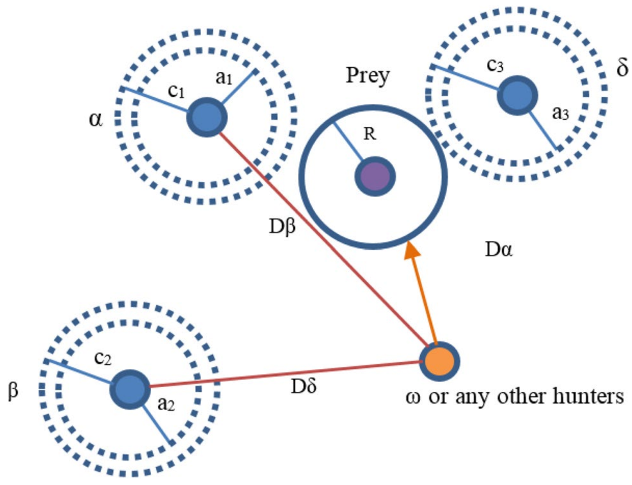

Fig. 1 Particle position updating in GWO algorithm

$$\vec{X}(t+1) = \vec{X_p}(t) - \vec{A}.\vec{D} \tag{2}$$

In these equations, t shows the number of current iterations. A and C are coefficients vectors, $\vec{X}_p$ the hunting position vector, and x the position vector of a grey wolf. D is the wolf's distance from the prey or the current position distance from the optimal response. Equation 2 is the new position of the wolf after moving towards the target. Vectors A and C are calculated by Eqs. 3 and 4 [54]:

$$\vec{A} = 2\vec{a}.\vec{r} - \vec{a} \tag{3}$$

$$\vec{C} = 2.\vec{r} \tag{4}$$

Vector decreases linearly from 2 to 0 during the iteration period in both the exploration and exploitation phases and r is a random vector from 0 to 1. Given the stochastic nature of r_1 and r_2 vectors, the wolves are allowed to reach any position between the points shown in Fig. 1. Thus, a grey wolf can change its position within the space that encompasses the prey at random using Eqs. 5 and 6. The same concept can be extended to n-dimensional search space. In this case, the grey wolves move around the cubes around the best solution. In Eq. 5, variable D shows the spatial distance of the alpha, beta, and delta wolves from the prey position or variable X. In Eq. 6, variables X_1, X_2, and X_3 are the new positions of the alpha, beta, and delta wolves are after changing the location and approaching the prey. Equation 7 calculates the new location of the hunter based on the mean of the three newer locations of alpha, beta, and delta wolves.

Grey wolves' hunting is usually guided by alpha. Beta and delta sometimes take part in hunting as well. We save three of the best solutions obtained and force the other search agents according to Eq. 7 to update their position according to the best search factors to model this behavior. Figure 1 shows how

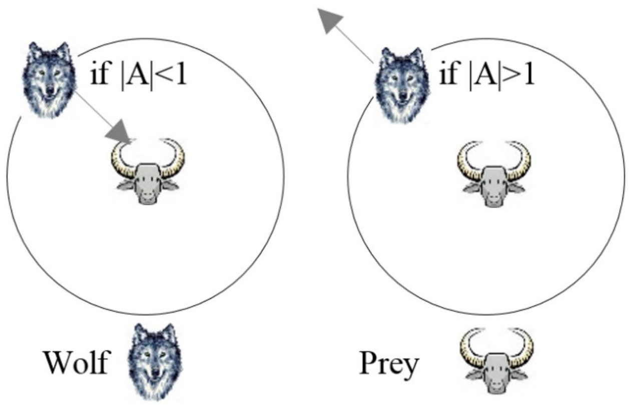

Fig. 2 Exploration phase versus exploitation

to update the search agent position in 2D space. According to Fig. 1, alpha, beta and delta estimate the hunting position, and other wolves randomly update their position around the hunting area. In Fig. 1, alpha, beta, delta, and omega wolves are shown as circles. D_α, D_β, and D_δ show the hunter's distance from other wolves. Variables a and c are the radius of spatial variations of alpha, beta, and delta particles and can be calculated through Eqs. 3 and 4. Variable R is the radius of prey locations change.

$$\overrightarrow{D_{\alpha|\beta|\delta}} = \left|\overrightarrow{C_{1|2|3}}.\vec{X}_{\alpha|\beta|\delta} - \vec{X}\right| \tag{5}$$

$$\begin{aligned}\overrightarrow{X_1} &= \overrightarrow{X_\alpha} - \overrightarrow{A_1}.\overrightarrow{D_\alpha}\\ \overrightarrow{X_2} &= \overrightarrow{X_\beta} - \overrightarrow{A_2}.\overrightarrow{D_\beta}\\ \overrightarrow{X_3} &= \overrightarrow{X_\delta} - \overrightarrow{A_3}.\overrightarrow{D_\delta}\end{aligned} \tag{6}$$

$$\overrightarrow{X(t+1)} = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{7}$$

In the exploitation phase or prey attack, the grey wolves will attack if the prey stops. We reduce the value of a from 2 to 0 to model this. The value of A, depending on a, decreases as well. The decrease in the value of A from 1 makes the wolves attack the prey. To avoid trapping at a local minimum of this algorithm, it provides a search or exploration phase for the prey. The wolves are separated from each other in search of prey and work together to attack it. To simulate this divergence, we use vector A with random values greater than 1 or smaller than 1. Figure 2 shows this problem.

Another component affecting the exploration process is C value. The value of this random number vector is in the range [0, 2]. If the random value C is greater than 1, the prey position will affect the wolf and prey distance (variable D in Eq. 5). However, if this value is less than 1, the prey position will be less effective. This vector can be considered as the effect of obstacles that prevent approaching the prey in nature. The pseudo-code of this algorithm is given in Table 1. All variables like the number of algorithm iterations, random variables A and C, the population of wolves, and the parameter a are initialized. In each iteration, which is shown by variable t, the wolves' population is randomly generated and the fit function is run for each case. In each iteration among the population, the best wolves are identified by alpha, beta, and delta based on their identified fitness. Then, the new position of the hunter wolves is determined based on the average location values of the top three wolves and all parameters and spatial vectors are updated. The best position of the wolves (X_α) is recorded as the response in each iteration.

Table 1 Pseudocode of the GWO algorithm

```
Set the initial values of the population size $n$, parameter $a$, coefficient vectors A and C, and the maximum number of iterations Maxiter.
Set t = 0.
for (i = 1 : n) do
  Generate an initial population of Xi(t) randomly.
  Evaluate the fitness function of each search agent (solution) f (Xi).
end for
Assign the values of the 1st, 2nd, 3rd best solution $X\alpha$, $X\beta$, $X\delta$ respectively.
repeat
for (i = 1 : n) do
Update each search agent in the population as shown in Eq. (7).
   Decrease the parameter $a$ from 2 to 0.
   Update the coefficients A, C as shown in Eq. (3) and (4), respectively.
   Evaluate the fitness function of each search agent (vector) $f(Xi)$.
end for
Update the vectors $X\alpha$, $X\beta$, and $X\delta$.
Set t = t + 1.
until (t ≥ Maxiter ). (Termination criteria are satisfied)
Produce the best solution $X\alpha$.
```

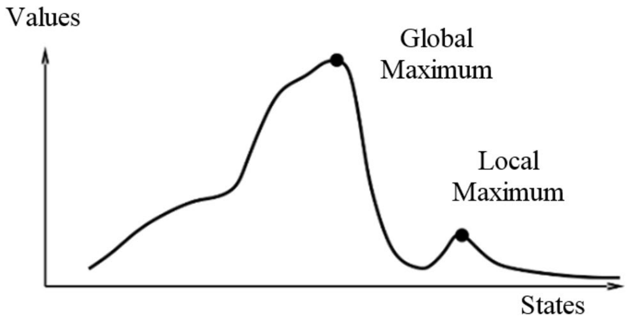

Fig. 3 Hill–Climbing problem

4 Proposed algorithm

In the GWO algorithm, the population moves towards the optimal responses—alpha, beta, and delta wolves move. In each iteration, the optimal particles are identified based on the fitness function and the best particle is called alpha. Each dimension of the new location is equal to the corresponding dimension mean of the superior particles, fully explained in Sect. 2 in Eqs. 1–7. The movement of the particles at each stage is done regardless of the degree of fitness; i.e., non-greedily.

In the proposed algorithm, the number of steps of a particle can change without an increase in the degree of fitness. Every change in the fitness of the new location is compared to the best location it used to be. If there are no definite steps to improve, the particle is returned to the last optimal response [55]. This is well shown in Fig. 3. As is seen in this figure, after reaching the global optimum, it continues to explore several steps until it reaches a better response; however, it returns to the general optimum after not finding appropriate responses.

The basic GWO algorithm does not perform well in the exploration of global optimizations [21]; thus, we used 10 functions according to chaos theory such as circular, Gaussian and logistic instead of conventional stochastic functions to reduce this effect and increase the efficiency of the proposed algorithm [56]. These functions are used to create numbers between [0, 1] as seen in Table 2. The initial value of all random numbers is considered 0.7 [57].

Overall, chaos, deterministic and quasi-random functions on dynamic and nonlinear systems are non-periodic, non-convergent, and finite. Mathematically, chaos functions are a random deterministic dynamic system. Chaos maps different from alternate mathematical functions can be used to use these functions in the optimization algorithm. Since the last decade, these functions have widely been focused on optimization because of their dynamic behavior that helps optimization algorithms in dynamic and more general discovery. Most importantly, chaos functions are used in real-world applications to make the algorithms applied. The results show that using chaos theory-based functions is effective to avoid being trapped in local optimum and to increase convergence speed. The implementation of some of these functions can be seen in Fig. 5. A random chaos function is used in each iteration of the GWO algorithm. Table 3 shows the pseudo-code for the proposed algorithm. The flowchart of the algorithm is also shown in Fig. 4 for more clarity.

Chaotic random numbers have a good effect on the convergence rate of the algorithm. Maps of the chaos functions generate random numbers within a permissible range. These numbers are initially predictable for a very short time and are random for a long time then.

5 Simulation and results

We used 23 standard mathematical optimization functions presented as CEC 2005 to compare the GWO algorithm and the proposed algorithm [56, 58, 59]. These benchmark functions have been selected as single exponential, multi-exponential, and finite-dimensional with varying hardness levels. The simulation and the resulting numerical results are performed in MATLAB 2017. The simulator uses a computer with a Core i7 processor with 2 GHz processing power and 4 GB of main memory.

We run the proposed algorithm 10 times over the relevant functions and obtain the maximum, minimum, median, and mean of the iterations as are shown in Tables 5, 7, and 9. All the results shown in this paper are based on the IEEE CEC 2005 approved format. In these tables, the results are better distinguished by the thick pen. Each time the algorithm is fully run, 1000 searches are performed. We have a population size of 30 and each response is assumed to be a set of 30. The population and computational power of the compared algorithms are considered similar to have a correct and fair comparison.

5.1 Unimodal benchmark functions

Figure 6 is a three-dimensional drawing of these benchmark functions. Moreover, the cost functions along with the dimensions, ranges, and minimum inputs related to the single exponential benchmark functions are shown in Table 4. In Tables 4, 6, and 8, n shows the number of x members. We used an array of length 30 for each particle.

These functions are suitable for measuring the exploitation process. Table 5 shows the statistical results (mean, median, minimum, and maximum) of the basic GWO algorithm, the proposed algorithm, and several new algorithms on unimodal exponential functions. Figure 7 shows the convergence graph for the best response to the algorithms. In

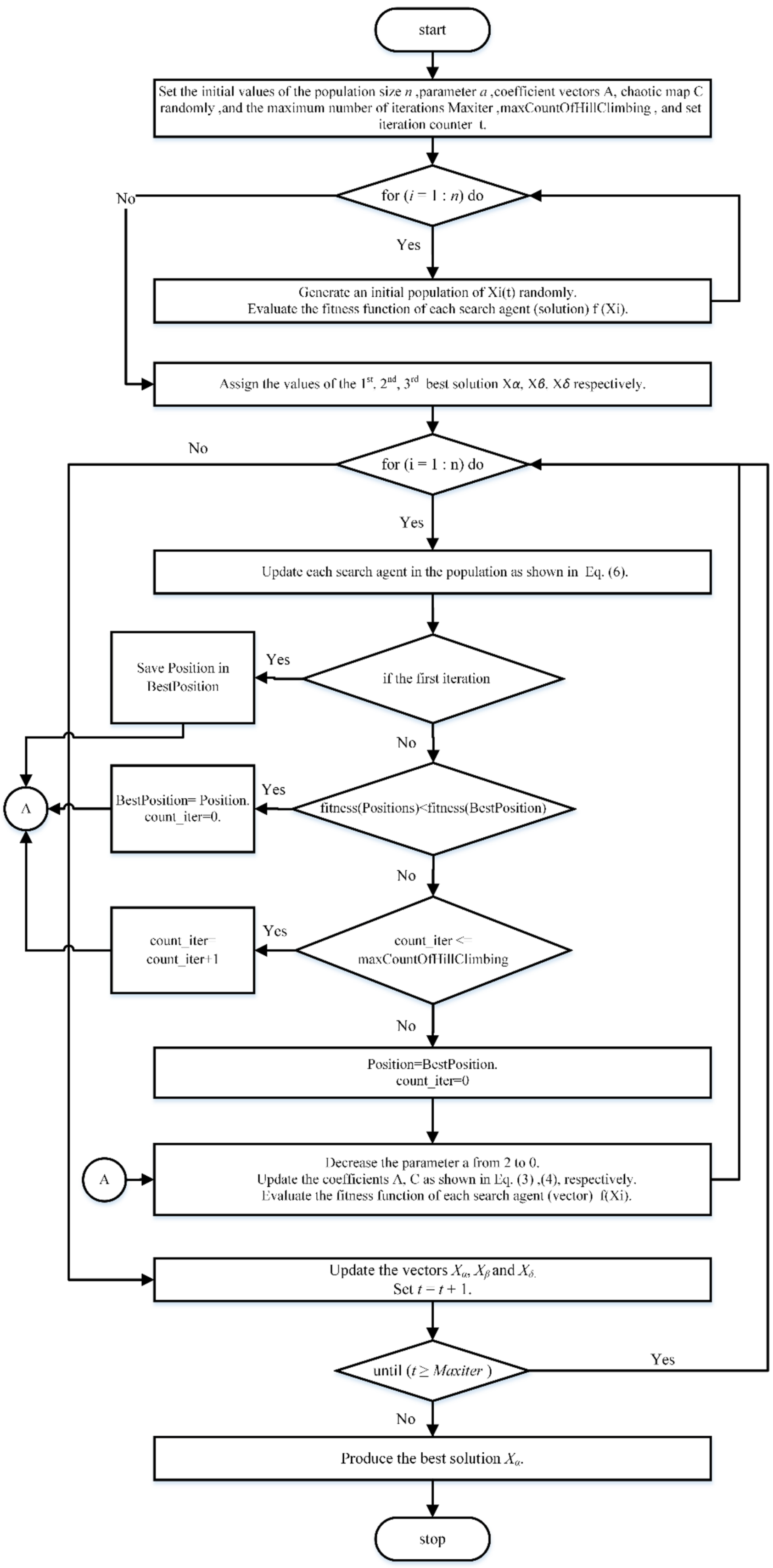
Fig. 4 Flowchart of the proposed HCGWO algorithm
start
Set the initial values of the population size n ,parameter a ,coefficient vectors A, chaotic map C randomly ,and the maximum number of iterations Maxiter ,maxCountOfHillClimbing , and set iteration counter t.
for (i = 1 : n) do
Yes
No
Generate an initial population of Xi(t) randomly. Evaluate the fitness function of each search agent (solution) f (Xi).
Assign the values of the 1st, 2nd, 3rd best solution Xα, Xβ, Xδ respectively.
for (i = 1 : n) do
Update each search agent in the population as shown in Eq. (6).
if the first iteration
Save Position in BestPosition
fitness(Positions)<fitness(BestPosition)
BestPosition= Position. count_iter=0.
count_iter <= maxCountOfHillClimbing
count_iter= count_iter+1
A
Position=BestPosition. count_iter=0
Decrease the parameter a from 2 to 0. Update the coefficients A, C as shown in Eq. (3) ,(4), respectively. Evaluate the fitness function of each search agent (vector) f(Xi).
Update the vectors Xα, Xβ and Xδ. Set t = t + 1.
until (t ≥ Maxiter)
Produce the best solution Xα.
stop

Table 2 Chaotic maps

No.	Name	Function
1	Chebyshev	$V_{b+1} = cos\left(bcos^{-1}\left(V_b\right)\right), b=1\ldots100$
2	Circle	$V_{b+1} = V_b + d - \left(\frac{C}{2\pi}\right)sin\left(2\pi V_b\right)mod(1), C=0.5, d=0.2$
3	Gauss	$V_{b+1} = \begin{cases} 0 & V_b = 0 \\ \frac{1}{V_b mod(1)} & otherwise \frac{1}{V_b mod(1)} = \frac{1}{V_b} - \left[\frac{1}{V_b}\right] \end{cases}$
4	Iterative	$V_{b+1} = sin\left(\frac{C\pi}{V_b}\right), C=0.7$
5	Logistic	$V_{b+1} = CV_b\left(1 - V_b\right), C=4$
6	Piecewise	$V_{b+1} = \begin{cases} V_b/P & 0 \le V_b < P \\ V_b - P/0.5 - P & P \le V_b < 1/2 \\ 1 - P - V_b/0.5 - P & 1/2 \le V_b < 1 - P \\ 1 - V_b/P & 1 - P \le V_b < 1, P = 0.4 \end{cases}$
7	Sine	$V_{b+1} = \frac{C}{4}\sin\left(\pi V_b\right), C = 4$
8	Singer	$V_{b+1} = \mu\left(7.86V_b - 23.31V_b^2 + 28.75V_b^3 - 13.3V_b^4\right), \mu=1.07$
9	Sinusoidal	$V_{b+1} = CV_b^2\sin\left(\pi V_b\right)$
10	Tent	$V_b=0.6$ $V_{b+1} = \begin{cases} V_b/0.7 & V_b < 0.7 \\ 10/3\left(1 - V_b\right) & V_b \ge 0.7 \end{cases}$

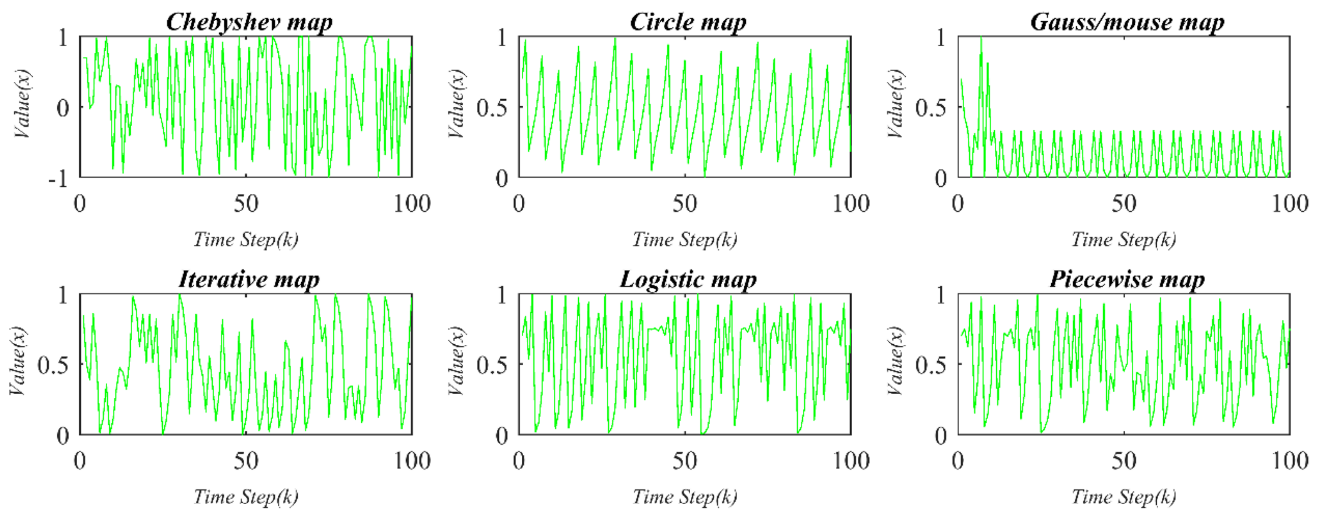

Fig. 5 Chaotic maps implementation result

Figs. 7, 9, and 11, the number of iterations is shown 80 times less, so the number of iterations in this figure is 12.5.

The results in Tables 5, 7 and 9 show the comparison of the proposed algorithm (HCGWO) with GWO Algorithms [54], Chaos Theory-based GWO (CGWO) [21], Hill-climbing Improved GWO (HGWO), PSO, ALO [13], multi-verse optimizer (MVO), and MFO algorithms [15]. Table 5 shows the statistical results of unimodal functions for the mentioned algorithms. According to the results of Fig. 7, the proposed algorithm exploitation stage performs better than the other algorithms. The results of the two improved algorithms of the other grey wolves are better than the other algorithm except for F6 function. According to Table 5, the results of the proposed algorithm mean, median, minimum, and maximum in all single unimodal functions except F6 are better than other algorithms. In F6 function, the result of the PSO algorithm is better as well. In the proposed algorithm, because of using the hill-climbing problem, the exploitation stage has improved significantly compared to the basic algorithm.

5.2 Multimodal benchmark functions

These functions evaluate the exploration stage and the ability to avoid the local optimum of the search algorithm. In CEC 2005 test problems, functions F8–F16 are multimodal.

Table 3 Pseudocode of proposed HCGWO algorithm

```
Set the initial values of the population size n, parameter a, coefficient vectors A and Set the initial value of the
chaotic map C randomly, and the maximum number of iterations Maxiter, maxCountOfHillClimbing.
Set t := 0.
for (i = 1 : n) do
  Generate an initial population of Xi(t) randomly.
  Evaluate the fitness function of each search agent (solution) f (Xi).
end for
Assign the values of the 1st, 2nd, 3rd best solution $X\alpha$, $X\beta$, $X\delta$ respectively.
repeat
for (i = 1 : n) do
    Update each search agent in the population as shown in Eq. (6).
        if the first iteration
          Save Position in BestPosition
          else if(fitness(Positions)<fitness(BestPosition))
              BestPosition= Position.
              count_iter=0.
              else if count_iter <= maxCountOfHillClimbing
                  count_iter=count_iter+1.
                  else
                      Position=BestPosition.
                      count_iter=0
                  end if
              end if
        end if
    Decrease the parameter a from 2 to 0.
    Update the coefficients A, C as shown in Eq. (3) ,(4), respectively.
    Evaluate the fitness function of each search agent (vector) f(Xi).
end for
Update the vectors $X\alpha$, $X\beta$, and $X\delta$.
Set t = t + 1.
until ($t \geq$ Maxiter ). (Termination criteria are satisfied)
Produce the best solution $X\alpha$.
```

The cost functions related to the multimodal benchmark functions along with the dimensions, range, and minimum input are shown in Table 6. Figure 8 is the three-dimensional diagram of the multimodal benchmark functions.

For a spatial vector with a length of 30 results, the mean of the proposed algorithm is less than the other algorithms. The statistical results for the multimodal functions are shown in Table 7. The results in all multimodal functions are better for the proposed algorithm except for F8 function that has a better mean and the best state for the moth algorithm. However, the result of the proposed algorithm is better than basic grey wolves, PSO chaos theory based GWO algorithm, and grey wolves with hill-climbing problem in this function. According to Fig. 9, the proposed algorithm exploration step is better than other algorithms. The chaos theory-based GWO algorithm performs better in F10, F11, and F15 functions than the basic GWO algorithm. The improved GWO algorithm with hill-climbing problem also performs better than the basic grey wolves in functions F9, F10, F12, F13, and F14. In the other multimodal functions, the results of the basic GWO algorithm are better than the chaos-theory based GWO algorithm. Because of using chaotic functions, the proposed algorithm exploration stage performs better than other algorithms.

5.3 Constrained multimodal benchmark functions

Constrained multimodal test problems to show the ability to avoid being trapped in local optimum and the balance between exploration and exploitation stages. Functions F17–F23 are constrained multimodal. The dimensions of these problems differ as shown in Table 8. Figure 10 is a three-dimensional drawing of these benchmark functions. According to the statistical results of Table 9, in these functions, the proposed algorithm is better than the other two algorithms with little difference. Figure 11 shows the results of the algorithms close to these functions. The results of the convergence diagram of functions F19–F23 were very similar to each other; thus, they have not been shown. Overall, the results of exploration, exploitation,

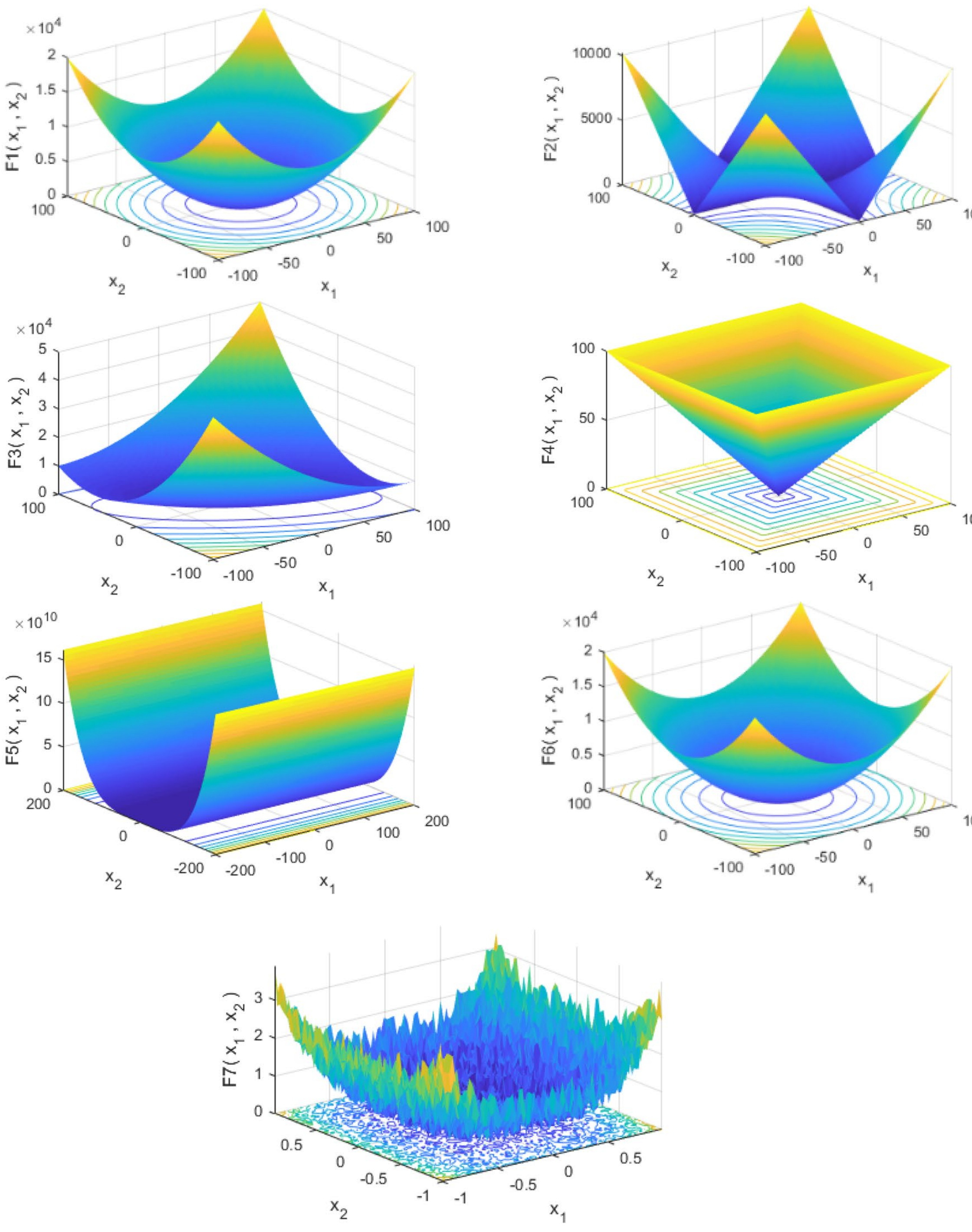

Fig. 6 3-D versions of unimodal benchmark functions

and local optimum avoidance steps of the proposed algorithm are better than or similar to other algorithms. The changes applied to the proposed algorithm have improved the relative balance between the exploration and exploitation steps.

6 Workflow scheduling in green cloud computing

According to the previous section, the HCGWO algorithm produces better compared to other optimization algorithms on various benchmark functions. This algorithm is designed to solve continuous optimization problems. In this section,

Table 4 Unimodal benchmark functions

Function	Dim	Range	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[− 100, 100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[− 10, 10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	[− 100, 100]	0
$f_4(x) = max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$	30	[− 100, 100]	0
$f_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right]$	30	[− 30, 30]	0
$f_6(x) = \sum_{i=1}^{n} \left([x_i + 0.5] \right)^2$	30	[− 100, 100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	[− 1.28, 1.28]	0

we will solve the workflow scheduling problem using the proposed algorithm. We can use the discrete environment optimization algorithm to solve this problem. The solutions must be binary to develop binary HCGWO. Hence, various conversions are required to realize this goal. Using transfer functions is one of the efficient ways to convert continuous optimizer to binary. Transfer functions are simple, fast, and low cost with simple implementation as well [60, 61].

In this paper, four S-shaped (S1–S4) and V-shaped (V1–V4) transfer functions are used to convert continuous HCGWO to binary. Table 10 shows the mathematical definition of these transfer functions. The images of these functions are shown in Fig. 12 as well.

Binary hill-climbing chaotic GWO (BHCGWO) algorithm could search binary search space in the light of using the transfer function. In this algorithm, the wolves' location changes in two stages. In the first step, the BHCGWO algorithm updates the location of wolves like the proposed HCGWO algorithm. This new location is in continuous space. In the second step, the transfer functions are used to convert the new location to a possible value. Wolf's new location is updated by Eqs. 8 and 9.

Thus, we convert the wolf's location into a binary form. In the set of S-shaped functions, the BHCGWO algorithm updates the location of the wolves according to Eq. 8.

$$X_i^d(t+1) = \begin{cases} 1 \ \ if\ rand(0,1) < T\left(\Delta X_i^d(t+1)\right) \\ 0 \ \ otherwise \end{cases} \tag{8}$$

In this equation, $T(x)$ is a S-shaped transfer function, rand(0,1) is a random number between [0,1], x the wolf's location, i the wolf number in the population, d is dimension and t is the current iteration number. Unlike the S-shaped transfer function, the V-shaped transfer function does not restrict the search agent to the interval [0,1]. Equation 9 shows updating the wolf's location with V-shaped transfer functions.

$$X_i^d(t+1) = \begin{cases} \neg X_i^d(t) \ \ if\ rand(0,1) < T\left(\Delta X_i^d(t+1)\right) \\ X_i^d(t) \ \ otherwise \end{cases} \tag{9}$$

In this equation, $T(x)$ is the S-shaped transfer function, rand(0,1) random number between [0,1], x wolf location, i wolf number in the population, d dimension, t current iteration number, and $\neg X$ supplement X. Table 11 shows the proposed binary algorithm.

We simulated the workflow problem in the Cloudsim environment and compared it with heterogeneous earliest finish time (HEFT), DHEFT, PSO, GWO, and CGWO algorithms to evaluate the proposed algorithm. The HEFT scheduling algorithm is a popular scheduling algorithm to minimize the execution time of tasks in the workflow. This algorithm has two stages of task prioritization and the last task selection phase. Each task is assigned to a processor with fewer EFTS for that task [34]. The distributed DHEFT or HEFT algorithm uses the distributed concept and better utilizes the concept of virtual machine accessibility level to better map tasks to the virtual machine [35]. The workflow scheduling and evaluation criteria will be discussed later on.

The parallel workflow can be shown by a non-circular graph in Fig. 13 [62]. The task graph G = (N, E) consists of a set of N vertices and E edges. In this problem, N is a set of tasks and E is a set of available edges between tasks that show prioritized constraints. Each $E \in edge(i, j)$ edge shows n_i and n_j tasks that n_j task cannot start until n_i task is completed [63]. Tasks without an input edge are the starting tasks. The actual start time (AST) is calculated for each n_i node on P_k processor using Eq. 10 [64].

$$AST\left(n_i, P_k\right) = \max\left(EST\left(n_i, P_k\right), Avail\left(P_k\right)\right) \tag{10}$$

In this equation, EST[1](ni, Pk) is the start time of task n_i on P_k processor. $Avail(Pk)$ is the first time P_k processor is ready to perform the task. The earliest end time (EFT) of each node on P_k processor is calculated using Eq. 11 [65].

$$EFT\left(n_i, P_k\right) = AST\left(n_i, P_k\right) + W\left(n_i, P_k\right) \tag{11}$$

In Eq. 11, variable W is the time needed to process the task n_i on P_k processor. According to Eq. 12, the completion time of all tasks equals the end time of the output graph node [65]. In this practical example, the goal is to reduce the time all tasks are completed [66–71].

[1] Earliest Start Time.

Table 5 Comparison of optimization results obtained for the unimodal benchmark functions (F1-F7) in 10 iterations

	HCGWO	PSO	GWO	CGWO	HGWO	ALO	MVO	MFO
F1								
Average	**6.9468e−43**	9.1651e−05	1.4777e−27	2.5396e−33	8.8978e−35	0.0010336	1.2821	6008.3099
Median	**4.729e−43**	4.839e−05	2.5778e−28	2.3967e−34	6.1586e−35	0.00095777	1.2199	34.3623
Worst	**1.3044e−42**	0.00026612	6.3494e−27	1.0628e−32	1.9139e−34	0.001809	1.7088	20,001.507
Best	**1.5605e−43**	9.6984e−06	1.9799e−28	2.0596e−35	3.221e−36	0.00043216	1.0346	1.5905
F2								
Average	**7.5367e−26**	0.050295	6.9014e−17	3.6229e−20	1.8525e−21	35.5875	0.65587	25.8797
Median	**7.049e−26**	0.020011	5.5668e−17	3.5763e−20	1.469e−21	21.8016	0.64866	30.1091
Worst	**1.7323e−25**	0.18705	1.2032e−16	6.584e−20	3.9566e−21	88.0321	0.95114	30.3762
Best	**1.6865e−26**	0.008893	3.6789e−17	1.3372e−20	8.4795e−22	8.1671	0.47081	18.513
F3								
Average	**5.8122e−10**	79.4177	4.3846e−07	8.3761e−09	2.9211e−08	4086.4126	260.7753	17,436.4019
Median	**2.8485e−10**	71.0817	9.4586e−08	6.7131e−10	1.2189e−08	4149.6831	251.7489	13,793.9382
Worst	**2.1648e−10**	115.0266	1.3551e−06	2.7794e−08	8.5956e−08	5472.5221	460.8192	32,607.7329
Best	**1.3918e−11**	56.4176	4.0187e−08	1.9717e−10	4.0169e−10	2325.1234	94.394	8083.9653
F4								
Average	**3.9227e−11**	1.1884	2.4308e−07	3.1475e−09	1.0864e−08	17.8541	2.1266	66.2307
Median	**2.2074e−11**	1.2325	2.4182e−07	3.2281e−09	7.5578e−09	17.0726	2.2719	69.4952
Worst	**9.2415e−11**	1.5415	3.92e−07	4.0598e−09	2.5863e−08	23.3831	3.0227	71.6198
Best	**6.7287e−12**	0.78219	1.2131e−07	1.8432e−09	1.8626e−09	15.404	1.1682	59.7321
F5								
Average	**25.8923**	47.8821	26.7316	27.9859	25.936	209.9292	898.1287	19,100.8589
Median	**26.131**	27.5914	27.0831	27.9819	26.2218	171.3735	449.7371	732.6185
Worst	**27.1106**	88.57	28.1283	28.7999	27.8859	436.7595	3005.2074	90,288.0057
Best	**24.8969**	22.6936	26.1005	27.168	25.1845	81.8003	41.5783	464.6745
F6								
Average	0.15121	**0.00012322**	0.82173	2.1685	0.80149	0.0010195	1.3952	1991.5863
Median	6.5663e−05	**0.00014334**	0.64331	1.9932	0.96824	0.00076701	1.2963	12.6365
Worst	0.50181	**0.00026487**	1.4933	3.242	1.0031	0.0019705	1.8247	9900.9301
Best	3.1408e−05	**1.768e−05**	0.49516	1.2535	0.49862	0.00055122	0.86741	6.4579
F7								
Average	**0.0011435**	0.1972	0.0023893	0.0011597	0.0014318	0.3006	0.035581	1.0797
Median	**0.00063433**	0.22819	0.0024451	0.001091	0.0013036	0.25302	0.026729	0.7112
Worst	**0.0022767**	0.2717	0.0030328	0.0020419	0.0027826	0.54887	0.069679	2.7735
Best	**0.00032155**	0.10398	0.0012816	0.00050637	0.0006637	0.2007	0.022797	0.15342

$$makespan = max\left\{EFT\left(n_{exit}\right)\right\} \tag{12}$$

In cloud computing, the computational cost for each customer is calculated according to the length of time the resources are used. Indeed, this cost is obtained by the execution time of task t_i on VM_j virtual machine according to Eq. 13. The duration of the task is according to Eq. 14.

$$C_p\left(t_i\right) = ET_{t_i}^{VM_j} \times CostPerProcessingVMj \tag{13}$$

$$ET_{t_i}^{VM_j} = MI\left(t_i\right)/MIPS\left(VM_j\right) \tag{14}$$

In this equation, $MI(ti)$ is the number of instructions of request i and $MIPS(VM_j)$ is the number of millions of instructions that machine j executes per second. The cost of storage is calculated according to the time needed to store the task on the virtual machine according to Eq. 15. In this equation, $WT_{t_i}^{VM_j}$ according to Eq. 16 is the waiting time of request ti on VM_j virtual machine to provide the required resources.

$$C_s\left(t_i\right) = (ET_{t_i}^{VM_j} + WT_{t_i}^{VM_j}) \times CostPreStorageInVM_j \tag{15}$$

$$WT_{t_i}^{VM_j} = maxinput\left(t_i\right)/BW \tag{16}$$

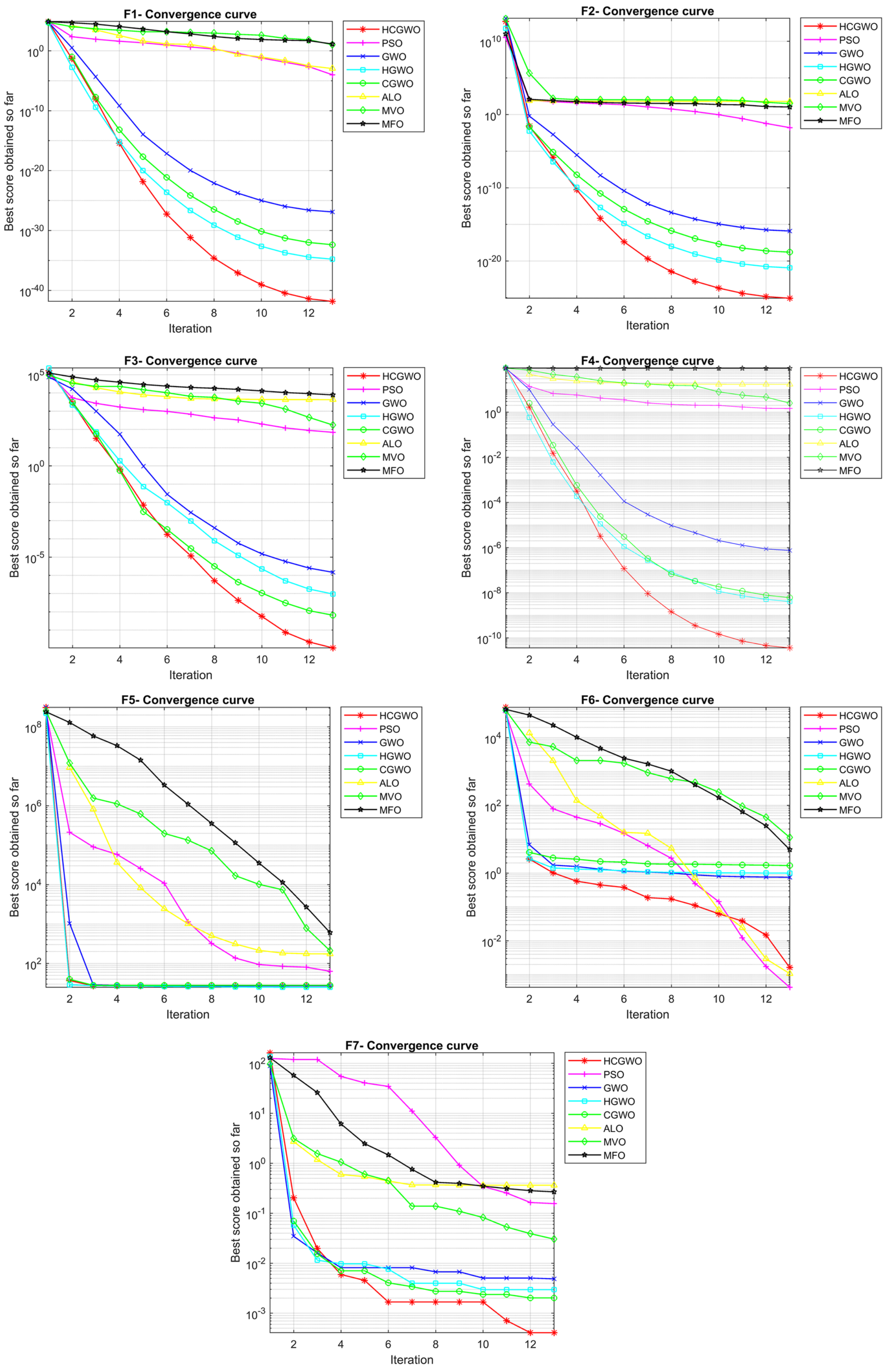

Fig. 7 Comparison of convergence curves of HCGWO and other algorithms in some of the unimodal benchmark functions

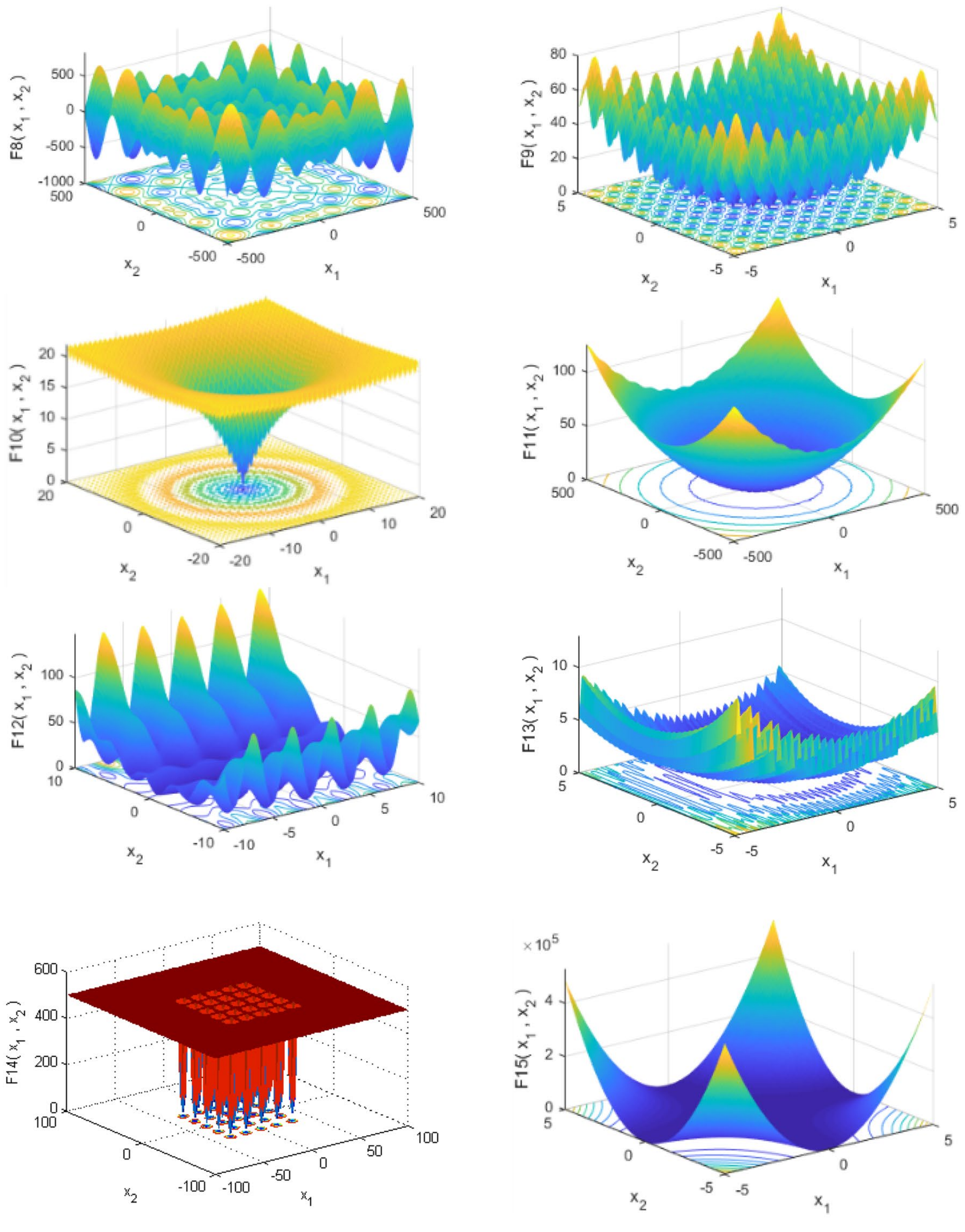

Fig. 8 3-D versions of multimodal benchmark functions

The cost of sending files for task t_i is calculated by Eq. 17. The total cost is equal to Eq. 18 [72].

$$C_T(t_i) = (\sum Output(t_i)/BW) \times CostPerTransfer \quad (17)$$

$$C_{total}(t_i) = C_p(t_i) + C_s(t_i) + C_T(t_i) \quad (18)$$

This paper considers the energy consumed in calculating tasks. Hence, the power consumption of task t_i on the virtual machine VM_{ij} is expressed with Eq. 19. In this equation, VE_{ij} is the hourly power consumption of machine VM_{ij}. Thus,

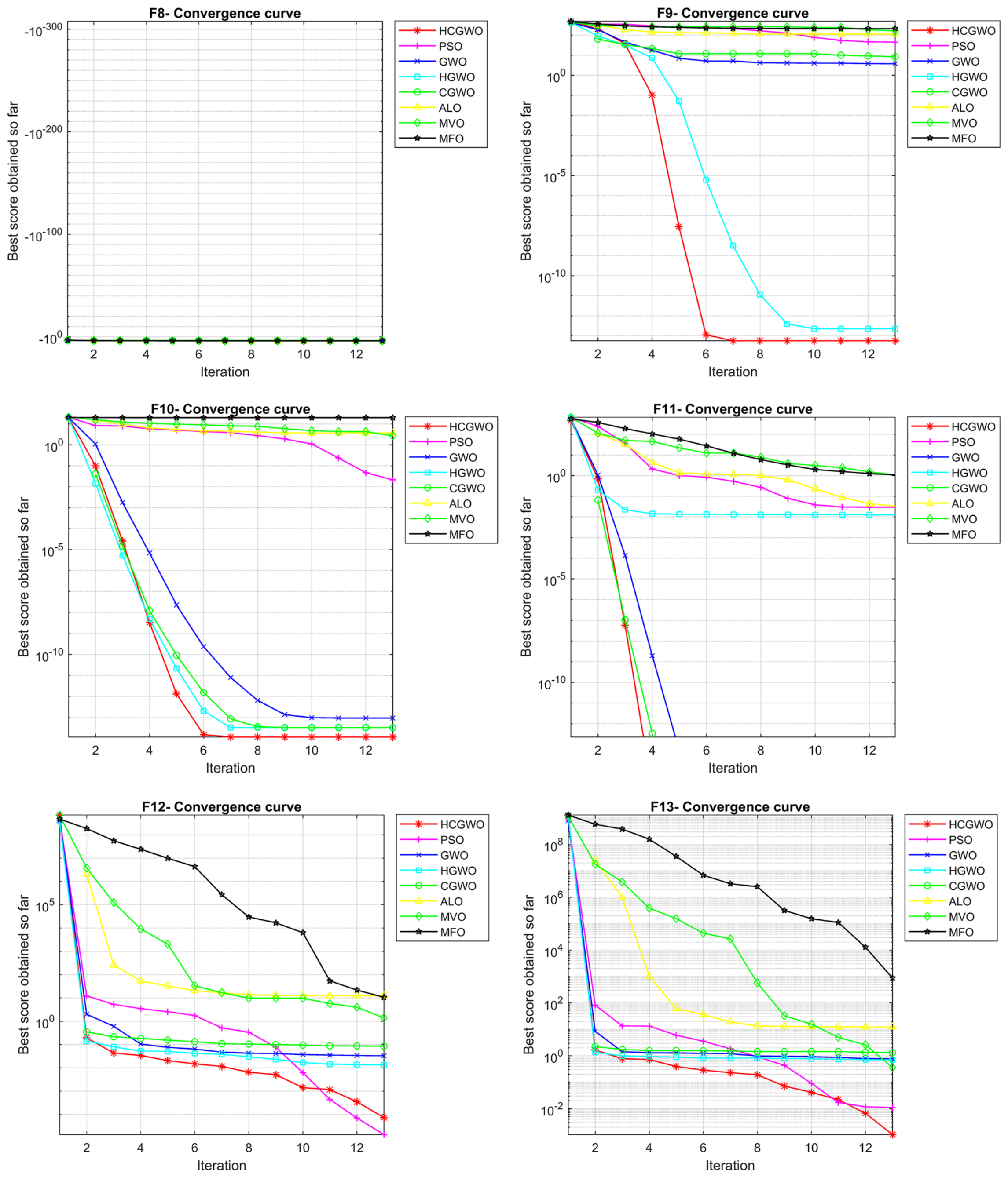

Fig. 9 Comparison of convergence curves of HCGWO and other algorithms in some of the multimodal benchmark functions

the energy consumption of a workflow equals the sum of the energy consumption of its workflow tasks according to Eq. 20 [73].

$$E(t_i, VM_{ij}) = ET_{t_i}^{VM_j} \times VE_{ij} \tag{19}$$

$$E(V) = \sum_{i=1}^{n} E(t_i, sched(t_i)) \tag{20}$$

We have several types of balanced and unbalanced workflows according to some scheduling papers [48, 74]. Table 12 shows the scientific workflows examined in this paper along with information like the number of vertices

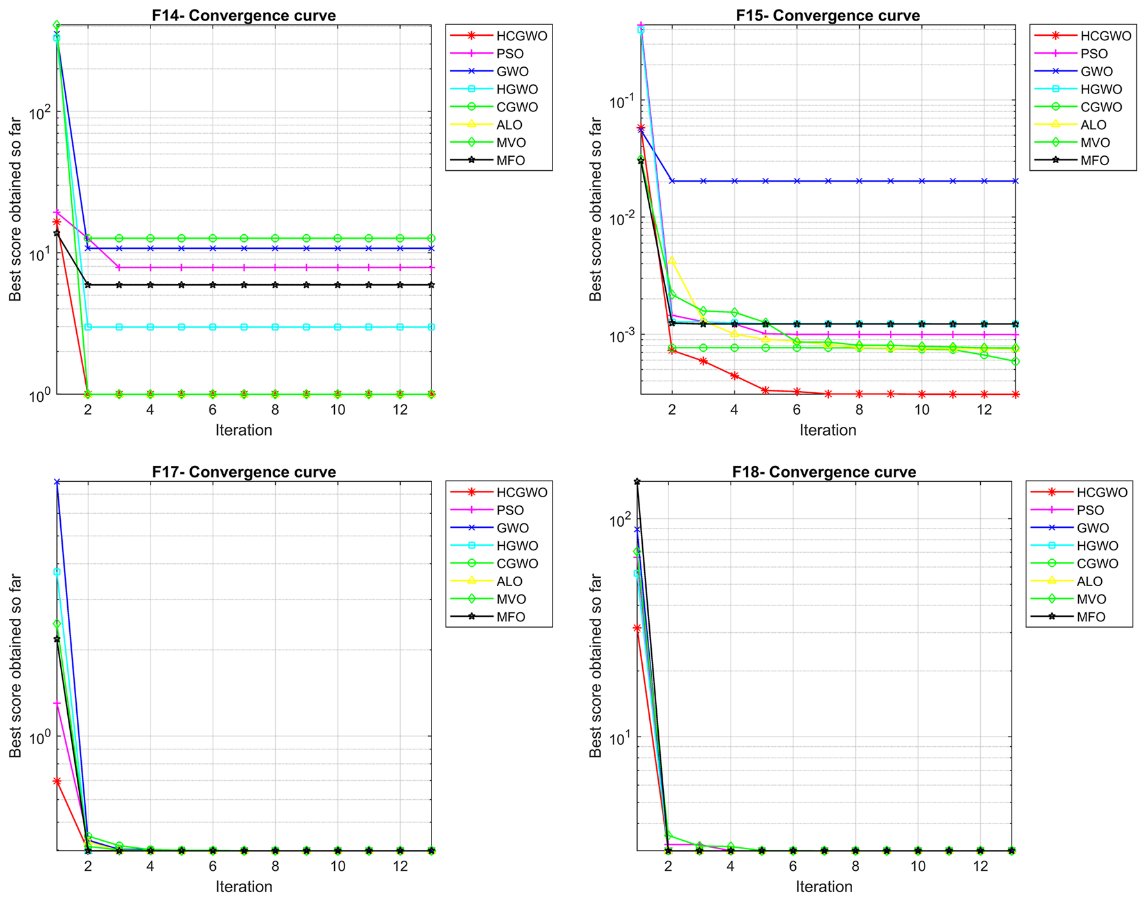

Fig. 9 (continued)

and edges, the minimum and maximum relevance of each edge in terms of bytes, the minimum and the maximum runtime [75]. Figure 13 is the graph of these workflows [2]. Experiments were done on 100 and 1000 vertices from four scientific workflows examined.

Epigenomics and Inspiral workflows are of the balanced type and Montage and Cybershake are of the unbalanced type. A balanced workflow has some pipelines that need similar services. However, they process various forms of services. The unbalanced workflow is more complex and has some parallel tasks that need various services.

The limited task distribution among the machines can meet the needs of users using the cloud. This section considers a particular type of cloud environment called green computing [76]. Green computing is known as environmentally friendly information technology. In other words, studying, designing, constructing, using, and disposition related systems and subsystems, so that they have minimal approval and exploitation from the environment. Reduction in energy consumption is one of the main goals of green computing, we can reach by designing efficient algorithms. In the rest of this section, the purpose is to evaluate the energy consumption of the proposed algorithm and examining the time and cost needed to perform the tasks [77].

As most scheduling presented does not reach the best possible response under all conditions, the paper tries to enhance the scheduling problem to some extent with an improved algorithm. We will compare the algorithms according to the three criteria - task completion time, cost, and energy. Figure 14 shows the 3D Pareto front diagram of the proposed algorithm. Figure 14 shows the results of the proposed algorithm for average workflows. Each chart has a number of blue and red dots where the red dots show the Pareto members optimally produced and the blue ones the ideal Pareto members produced. In these graphs, each dimension is one of the comparison benchmarks.

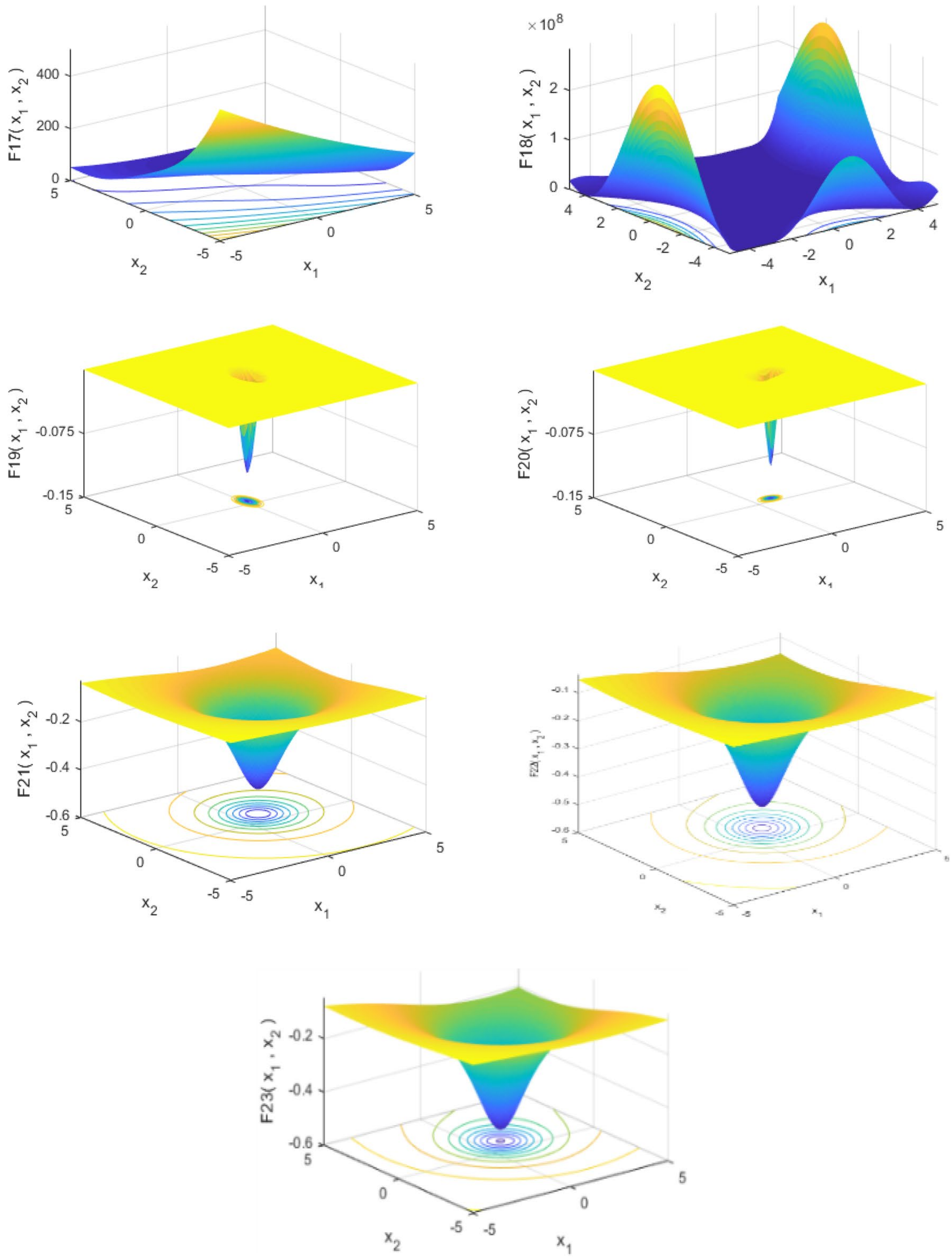

Fig. 10 3-D versions of fixed-dimension multimodal benchmark functions

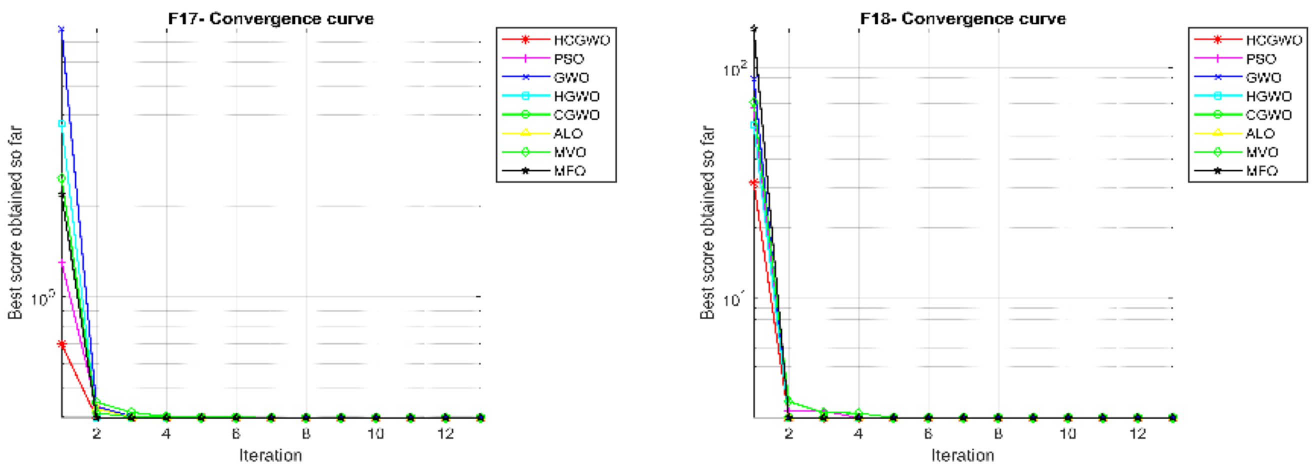

Fig. 11 Comparison of convergence curves of HCGWO and other algorithms in some of the fixed-dimension multimodal benchmark functions

Table 6 Multimodal benchmark functions

Function	Dim	Range	f_{min}
$f_8(x)=\sum_{i=1}^{n} -x_i sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	$[-500, 500]$	-418.9829×5
$f_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	30	$[-5.12, 5.12]$	0
$f_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+20+e$	30	$[-32, 32]$	0
$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$[-600, 600]$	0
$f_{12}(x)=\frac{\pi}{n}\left\{10\sin\left(\pi y_1\right)+\sum_{i=1}^{n-1}\left(y_i-1\right)^2\left[1+10sin^2\left(\pi y_{i-1}\right)\right]+\left(y_n-1\right)^2\right\}$	30	$[-50, 50]$	0
$+\sum_{i=1}^{n}u\left(x_i,10,100,4\right), y_i=1+\frac{x_i+1}{4}$			
$\mathrm{u}\left(x_i,\mathrm{a},\mathrm{k},\mathrm{m}\right)=\begin{cases}k\left(x_i-a\right)^m & x_i>a\\ 0 & -a<x_a<a\\ k\left(-x_i-a\right)^m & x_i<-a\end{cases}$			
$f_{13}(x)=0.1\left\{sin^2\left(3\pi x_1\right)+\sum_{i=1}^{n}\left(x_i-1\right)^2\left[1+sin^2\left(3\pi x_i+1\right)\right]+\left(x_n-1^2\right)\left[1+sin^2\left(2\pi x_n\right)\right]\right\}+\sum_{i=1}^{n}u\left(x_i,5,100,4\right)$	30	$[-50, 50]$	0
$f_{14}(x)=\sum_{i=1}^{n}\sin\left(x_i\right).\left(sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m}, m=10$	30	$[0, \pi]$	-4.687
$f_{15}(x)=\left[e^{-\sum_{i=1}^{n}\left(x_i/\beta\right)^{2m}}-2e^{-\sum_{i=1}^{n}x_i^2}\right].\prod_{i=1}^{n}cos^2 x_i, m=5$	30	$[-20, 20]$	-1
$f_{16}(x)=\left(\left[\sum_{i=1}^{n}sin^2\left(x_i\right)\right]-\exp\left(-\sum_{i=1}^{n}x_i^2\right)\right).\exp\left[-\sum_{i=1}^{n}sin^2\sqrt{\lvert x_i\rvert}\right]$	30	$[-10, 10]$	-1

Table 7 Comparison of optimization results obtained for the multimodal benchmark functions (F8-F15) in 10 iterations

	HCGWO	PSO	GWO	CGWO	HGWO	ALO	MVO	MFO
F8								
Average	−5989.1312	−5261.7293	−5717.0179	−5759.4266	−5515.5868	−6147.3232	−7928.8541	**−8787.1449**
Median	−5951.4982	−6487.3747	−6630.8358	−5630.793	−5935.1256	−5418.2808	−7957.8835	**−8996.8159**
Worst	−5182.9179	−3043.6746	−3636.7026	−4960.1357	−3950.0693	−5417.6748	−7147.4386	**−7589.3808**
Best	−7224.6131	−6566.2401	−6827.5529	−7068.7775	−6062.9433	−9016.2057	−8632.0256	**−9464.1723**
F9								
Average	**4.5475e−14**	72.1227	4.4907	1.95	0.89681	68.2553	101.2343	199.1907
Median	**2.2737e−13**	72.5695	0	0	0	66.6626	94.1638	194.1654
Worst	**0**	83.8416	17.1564	9.7501	4.484	86.5617	135.9787	35.0872
Best	**1.1369e−15**	59.9082	0	0	5.6843e−14	51.7423	77.2765	166.2726
F10								
Average	**3.9257e−14**	0.19691	1.0321e−13	1.581e−14	3.0731e−14	6.3755	2.0015	19.2137
Median	**3.9968e−14**	0.012861	1.0036e−13	1.5099e−14	3.2863e−14	3.0932	1.8154	19.8062
Worst	**4.3521e−14**	0.9314	1.1813e−13	1.8652e−14	3.2863e−14	12.8148	2.8642	19.9537
Best	**3.2863e−14**	0.0090414	9.3259e−14	1.5099e−14	2.5757e−14	2.1206	1.481	18.1572
F11								
Average	**0.0003269**	0.0082269	0.0075901	0.0040586	0.0077435	0.049923	0.89367	18.9906
Median	**0**	0.0098757	0	0	0.0079949	0.048667	0.91068	1.0154
Worst	**0.0057341**	0.0197	0.024532	0.020293	0.017934	0.061512	0.94242	90.9522
Best	**0**	1.3207e−05	0	0	0	0.037607	0.83485	0.95555
F12								
Average	**2.0493e−06**	0.0029994	0.045242	0.20223	0.017398	14.4259	2.4315	8.071
Median	**3.2293e−07**	4.0036e−06	0.039717	0.1248	0.013723	10.5685	2.253	7.1995
Worst	**8.0354e−06**	0.0084599	0.069629	0.56976	0.026658	24.55	3.4895	12.606
Best	**2.6278e−07**	1.6564e−06	0.025047	0.055229	0.013165	10.1767	1.9892	5.9687
F13								
Average	**0.0064788**	0.12063	0.54119	1.3162	0.63134	22.4864	0.13393	14.6478
Median	**0.00018974**	0.099803	0.66129	1.3342	0.62124	21.0918	0.12156	11.7512
Worst	**0.021046**	0.40289	0.78256	1.6261	0.79549	47.6724	0.20163	28.4673
Best	**1.7638e−06**	4.7981e−05	0.23354	0.83664	0.45384	0.10291	0.082835	2.7491
F14								
Average	**0.7447**	3.364	2.1885	7.2386	1.7916	2.1893	0.998	3.1715
Median	**0.9821**	1.992	2.9821	10.7632	0.998	1.992	0.998	2.9821
Worst	**1.7632**	6.9033	2.9821	12.6705	2.9821	2.9821	0.998	5.9288
Best	**0.998**	0.998	0.998	0.998	0.998	0.998	0.998	0.998
F15								
Average	**0.00090078**	0.00095386	0.0046906	0.0083382	0.0045018	0.0045018	0.008546	0.0010404
Median	**0.00072606**	0.00099349	0.0012232	0.00034895	0.0003075	0.00030749	0.0007665	0.00081231
Worst	**0.00162**	0.0010823	0.020363	0.020363	0.020363	0.020363	0.020363	0.0016554
Best	**0.00059093**	0.00072197	0.00030974	0.0003075	0.00030749	0.00030749	0.00055338	0.00052735
F16								
Average	**−1.0316**	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
Median	**−1.0316**	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
Worst	**−1.0316**	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
Best	**−1.0316**	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316

Table 8 Fixed-dimension multimodal benchmark functions

Function	Dim	Range	f_{min}
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 1$	2	[−5, 5]	0.398
$f_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	2	[−2, 2]	3
$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{i=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	[1, 3]	−3.86
$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{i=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	[0, 1]	−3.32
$f_{21}(x) = -\sum_{i=1}^{5} \left[(X - a_i)(X - a_i)^T + C_i\right]^{-1}$	4	[0, 10]	−10.153
$f_{22}(x) = -\sum_{i=1}^{7} \left[(X - a_i)(X - a_i)^T + C_i\right]^{-1}$	4	[0, 10]	−10.402
$f_{23}(x) = -\sum_{i=1}^{10} \left[(X - a_i)(X - a_i)^T + C_i\right]^{-1}$	4	[0, 10]	−10.536

According to the results shown in Figs. 15, 16, and 17, the proposed algorithm has relatively lower task completion time, cost, and power consumption compared to the basic algorithms. As is seen in Fig. 15, the proposed algorithm has managed to accomplish less time than other methods in different workflows. However, in the large-scale Montage workflow, the results of the proposed algorithm and the HEFT, PSO, GWO, and CGWO algorithms are close together. Figure 16 indicates the results for the cost of scheduling. The results of the proposed algorithm on medium-sized workflows have proven to be better than the other algorithms examined. The proposed algorithm scheduling cost in large-scale Epigenomics, Inspiral, and Cybershake workflows are better than other algorithms as is shown in Fig. 16. However, the results of this criterion on large-scale Montage workflow are better for the HEFT algorithm. Figure 17 is the results of the energy consumption of running algorithms on different workflows. The proposed algorithm on the Montage workflow with various dimensions has proper results as well. The results of this algorithm on mid-dimensional Epigenomics, Inspiral, and Cybershake workflows are better than other algorithms as well. In large-scale Epigenomics, Inspiral, and Cybershake workflows, the results of the presented algorithm are better than HEFT and DHEFT algorithms, whereas it yields improper results compared to PSO, GWO, and CGWO algorithms. The changes made on the proposed algorithm in task scheduling problems have a great effect on task completion time, cost, and energy consumption as a result of returning to the optimal states in case of failing to find proper responses.

7 Discussion

The innovation in the proposed algorithm is the improvement in search using hill-climbing problem and random numbers based on chaos theory. From the experiments performed in Sects. 5.1–5.3 on the benchmarks, it is shown that HCGWO performs better than the GWO and other optimization approaches. Besides, HCGWO implementation is easy and simple, and there is no effect on the fine-tuning of the parameters. The work performed in this paper shows the robustness of the HCGWO for all types of benchmark functions. There are several ways to check meta-heuristic optimization-based algorithms efficiently. Benchmark evaluation is a simple way that is widely used. However, it is not a perfect approach.

The use of chaos was one of the techniques used in metaheuristic algorithms to tune certain parameters. We added chaos to the basic GWO in the current work and created a chaotic GWO version. To validate the algorithm, ten different chaotic maps were used. The results suggest that the new algorithms are strengthened due to the implementation of deterministic chaotic signals instead of constant and/or random values. Statistical findings and the HCGWO's performance rates suggest that the tuned algorithms will significantly improve the reliability of the global optimality and also increase the consistency of the results. Any progress in such evaluation will undoubtedly provide insight into the working mechanism of chaotic metaheuristic algorithms and the confluence of metaheuristic algorithms with chaos.

Table 9 Comparison of optimization results obtained for the fixed-dimension multimodal benchmark functions (F17–F23) in 10 iterations

	HCGWO	PSO	GWO	CGWO	HGWO	ALO	MVO	MFO
F17								
Average	**0.39789**	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789
Median	**0.39789**	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789
Worst	**0.39789**	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789
Best	**0.39789**	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789
F18								
Average	**3**	3	3.0001	19.2	3	3	3	3
Median	**3**	3	3	3	3	3	3	3
Worst	**3**	3	3.0002	84	3	3	3	3
Best	**3**	3	3	3	3	3	3	3
F19								
Average	**−3.8628**	−3.8628	−3.8595	−3.8587	−3.8628	−3.8628	−3.8628	−3.8628
Median	**−3.8628**	−3.8628	−3.8592	−3.8581	−3.8628	−3.8628	−3.8628	−3.8628
Worst	**−3.8628**	−3.8628	−3.8565	−3.8549	−3.8628	−3.8628	−3.8628	−3.8628
Best	**−3.8628**	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628
F20								
Average	**−3.2982**	−3.2507	−3.2505	−3.298	−3.2497	−3.2503	−3.2979	−3.2132
Median	**−3.322**	−3.2031	−3.322	−3.322	−3.2029	−3.203	−3.322	−3.2031
Worst	**−3.2031**	−3.2031	−3.0839	−3.2022	−3.1992	−3.2019	−3.2017	−3.1345
Best	**−3.322**	−3.322	−3.322	−3.322	−3.322	−3.322	−3.322	−3.322
F21								
Average	**−10.1519**	−7.6272	−10.1505	−10.1485	−10.1510	−4.6514	−9.1424	−5.114
Median	**−10.1518**	−10.1438	−10.1506	−10.1478	−10.1511	−2.6829	−10.1528	−5.0552
Worst	**−10.1513**	−2.6305	−10.1482	−10.1468	−10.1505	−2.6829	−5.1007	−2.6305
Best	**−10.1525**	−10.1532	−10.1527	−10.1512	−10.1525	−10.1532	−10.1531	−10.1532
F22								
Average	**−10.4011**	−7.3453	−10.4004	−9.3445	−10.4011	−8.2768	−8.2848	−7.537
Median	**−10.4018**	−10.4029	−10.4006	−10.3993	−10.4003	−10.4029	−10.4025	−10.4029
Worst	**−10.3987**	−2.7519	−10.3993	−5.1237	−10.4013	−5.0877	−5.0876	−2.7519
Best	**−10.4027**	−10.4029	−10.4019	−10.4011	−10.4027	−10.4029	−10.4027	−10.4029
F23								
Average	**−10.5346**	−8.0958	−8.9124	−10.5339	−10.5323	−6.209	−8.9901	−8.1241
Median	**−10.5348**	−10.5364	−10.5348	−10.5341	−10.5326	−5.1285	−10.5361	−10.5364
Worst	**−10.5335**	−3.694	−2.4217	−10.5315	−10.5345	−2.4217	−2.8066	−3.8354
Best	**−10.5351**	−10.5364	−10.5358	−10.5354	−10.5350	−10.5364	−10.5361	−10.5364

Table 10 The utilized S-shaped and V-shaped transfer functions

S-shaped family	Transfer function	V-shaped family	Transfer function
S1	$T(x) = 1/\left(1 + e^{-2x}\right)$	V1	$T(x) = \left\lvert erf\left(\frac{\sqrt{\pi}}{2}x\right)\right\rvert$
S2	$T(x) = 1/(1 + e^{-x})$	V2	$T(x) = \lvert tanh(x)\rvert$
S3	$T(x) = 1/\left(1 + e^{\left(-\frac{x}{2}\right)}\right)$	V3	$T(x) = \left\lvert x/\sqrt{1 + x^2}\right\rvert$
S4	$T(x) = 1/\left(1 + e^{\left(-\frac{x}{3}\right)}\right)$	V4	$T(x) = \left\lvert \frac{2}{\pi}arctan\left(\frac{\pi}{2}x\right)\right\rvert$

A local search strategy works well in certain instances but usually fails with a large number of local maxima. Hill climbing is a local search strategy that aims to enhance a given solution by searching around the currently best-known solution in the solution space. Hill Climbing is therefore suggested as a method for the general case due to its capability to avoid local maxima and ease of use and implementation.

Ultimately, we compared the performance of the new algorithm with some of the famous existing algorithms in this regard and identified its strengths and weaknesses. The improvements made bring about an increase

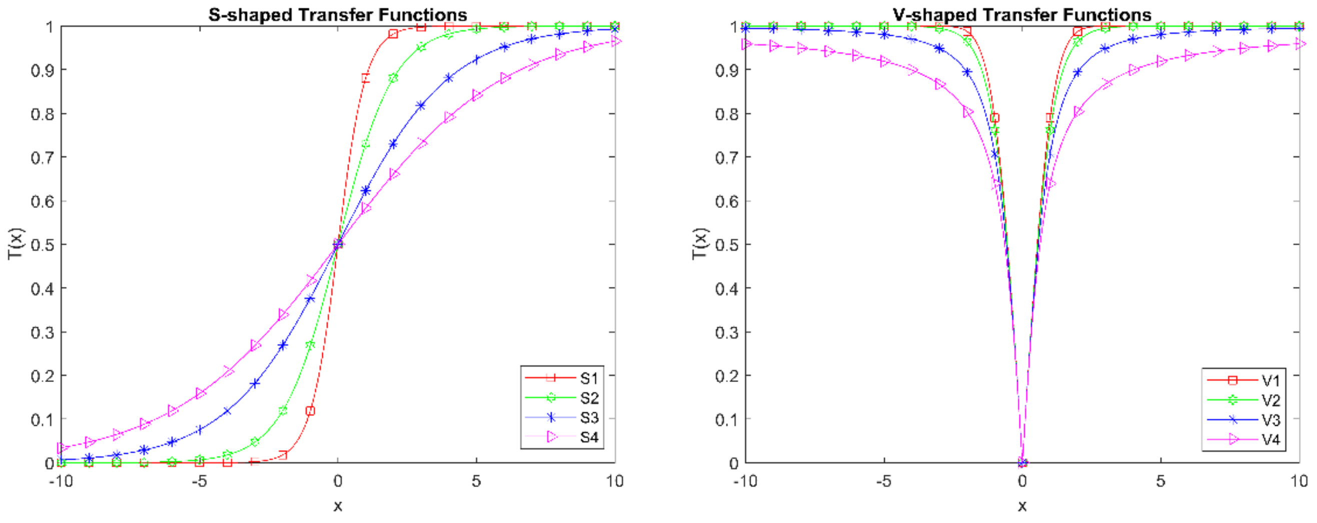

Fig. 12 Sample S-shaped and V-shaped transfer functions

Table 11 Pseudocode of proposed BHCGWO algorithm

```
Set the initial values of the population size $n$, parameter $a$, coefficient vectors A and Set the initial value of the chaotic map C randomly,
and the maximum number of iterations Maxiter, maxCountOfHillClimbing.
Set t := 0.
for (i = 1 : n) do
  Generate an initial population of Xi(t) randomly.
  Evaluate the fitness function of each search agent (solution) f (Xi).
end for
Assign the values of the 1st, 2nd, 3rd best solution $X\alpha$, $X\beta$, $X\delta$ respectively.
repeat
for (i = 1 : n) do
     Update each search agent in the population as shown in Eq. (6).
          if the first iteration
             Save Position in BestPosition
             else if(fitness(Positions)<fitness(BestPosition))
                   BestPosition= Position.
                   count_iter=0.
                   else if count_iter <= maxCountOfHillClimbing
                        count_iter=count_iter+1.
                        else
                             Position=BestPosition.
                             Calculate the probability using S-shaped or V-
                                  shaped transfer function.
                             Update a new position using (8) or (9).
                             count_iter=0
                        end if
                   end if
          end if
      Decrease the parameter $a$ from 2 to 0.
      Update the coefficients A, C as shown in Eq. (3) ,(4), respectively.
      Evaluate the fitness function of each search agent (vector) $f(Xi)$.
end for
Update the vectors $X\alpha$, $X\beta$, and $X\delta$.
Set t = t + 1.
until (t ≥ Maxiter ). (Termination criteria are satisfied)
Produce the best solution $X\alpha$.
```

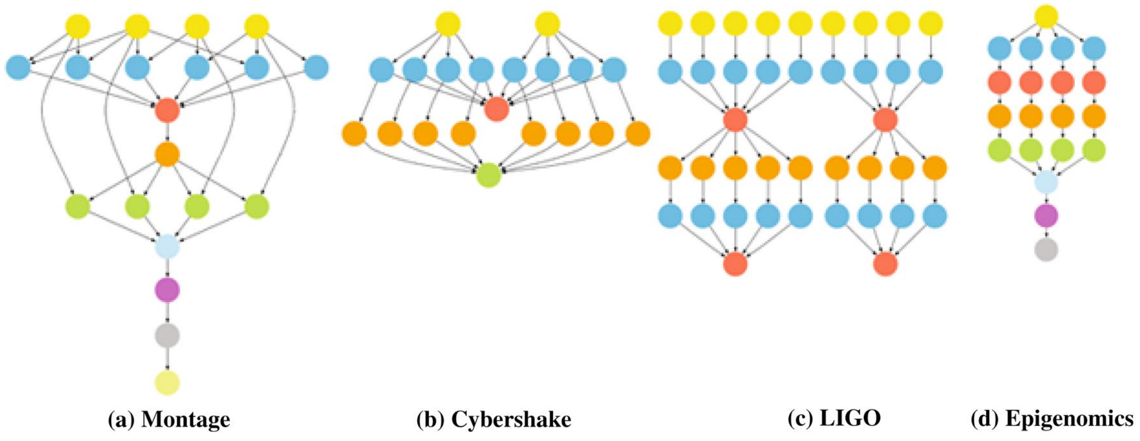

Fig. 13 Non-cyclic graph of scientific workflows

Table 12 Specification of scientific workflows

Workflow name	No. of vertices	No. of edges	Max com.	Min com.	Max. task runtime	Min. task runtime
Cybershake	100	190	10,300	102	203.78	0.34
Cybershake	1000	1994	1,003,000	1002	180.67	0.55
Epigenomics	100	124	10,300	25	23,471	0.02
Epigenomics	997	1242	997,000	246	27,775	0.01
Inspiral	100	250	10,300	24	670.45	4.25
Inspiral	1000	1486	1,003,000	230	689.97	4.23
Montage	100	250	10,300	63	13.85	0.83
Montage	1000	2652	1,003,000	663	99.53	2.52

in convergence speed and prevent trapping in local optimum by proper adjustment of the parameters. The paper used the proposed algorithm in an applied way for workflow scheduling in the green cloud computing environment. Thus, we made the proposed algorithm binary using transfer functions. The purpose was to minimize the cost and time of doing the tasks and workflows and to reduce energy consumption for having a green cloud environment. The results show a reduction in the energy consumed, the cost, and the time needed to perform tasks in some cases.

8 Conclusion

The purpose of the paper was to present a new version of the metaheuristic GWO algorithm to optimize the search capability in wolves hunting. Thus, 23 standard IEEE CEC2005 benchmark functions were used to evaluate the performance of the presented algorithm. The efficiency of the proposed algorithm proved to be better compared to the base algorithm and other algorithms. In the algorithm proposed for random numbers, we use several chaos functions and keep the best possible response for some determined

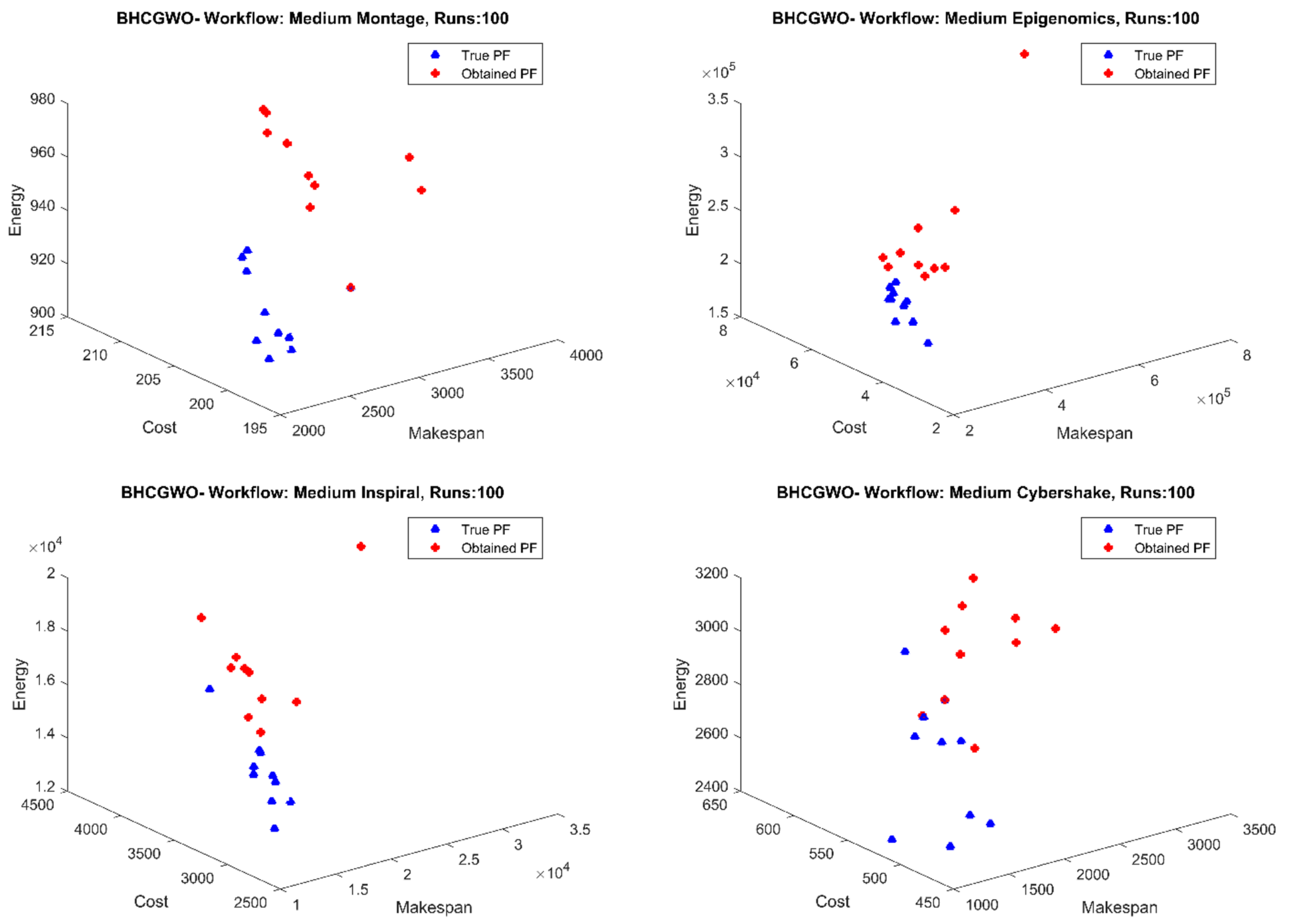

Fig. 14 Pareto front of scientific workflows in medium dimension

steps until we return to the last optimal response if no better responses are reached. This improvement led to increased convergence speed and prevented trapping in a local optimum. Moreover, the practical task scheduling problem was proposed on scientific workflows of various sizes and the performance of the binary version of the proposed algorithm was examined on the task scheduling problem to prove the applicability of the algorithm. The simulation results showed the reduction of runtime, cost, and energy for the proposed algorithm.

The proposed algorithm has opened a new path for enhancing leadership-based search capability. Other improvements could be proposed for this algorithm in the finite problems and infinite to multi-swarm multi-objective states.

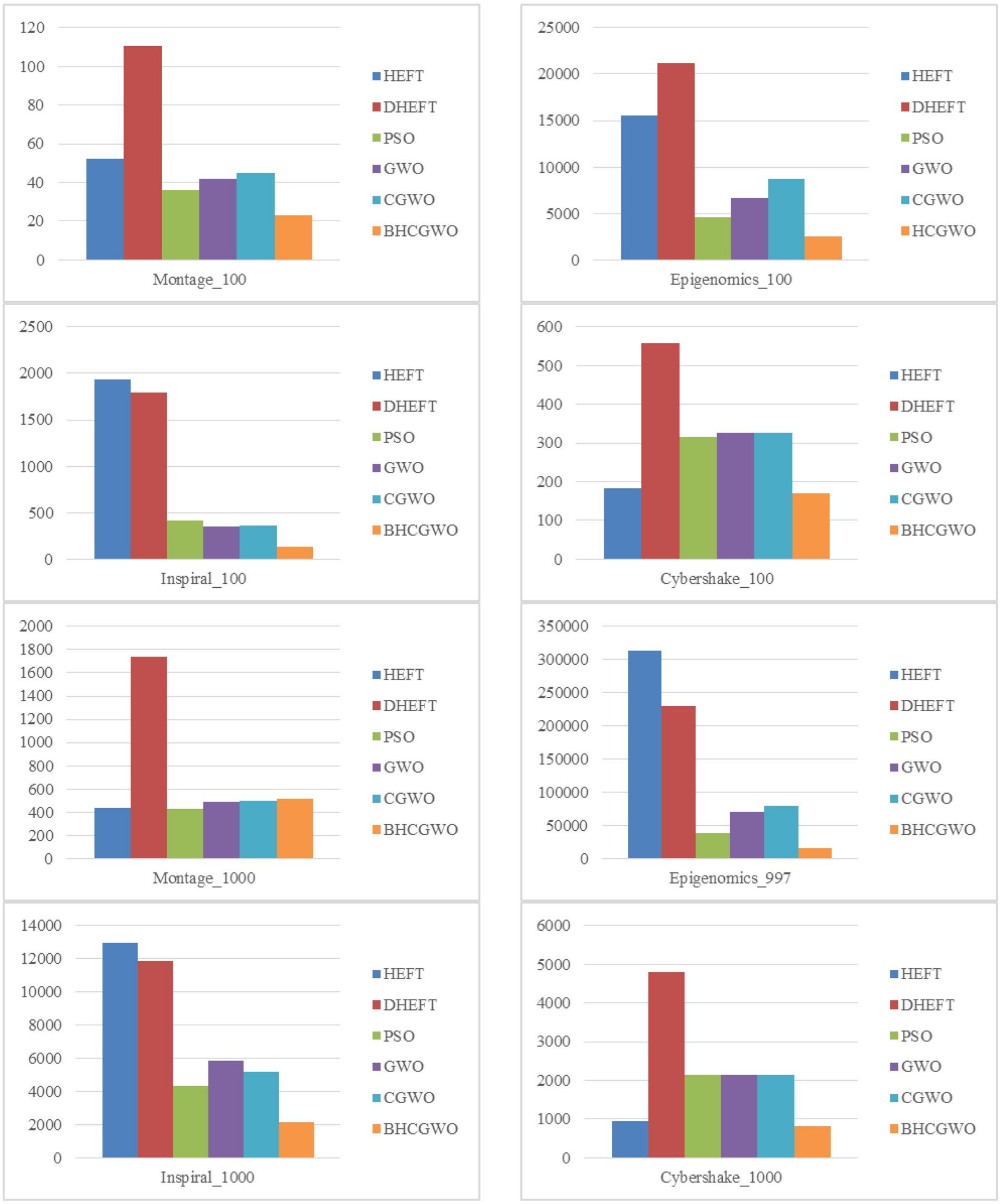

Fig. 15 Comparison of Makespan

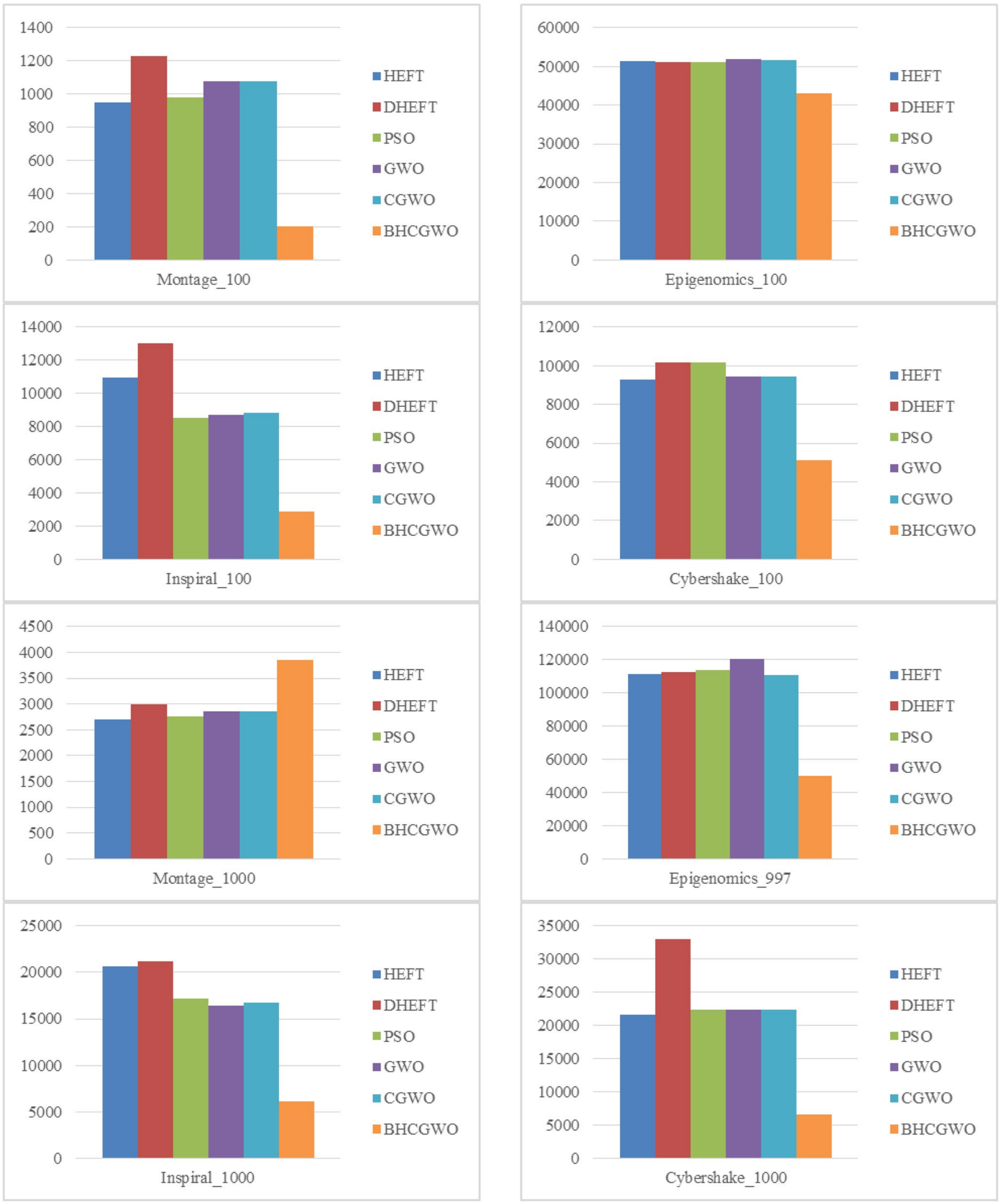

Fig. 16 Comparison of cost

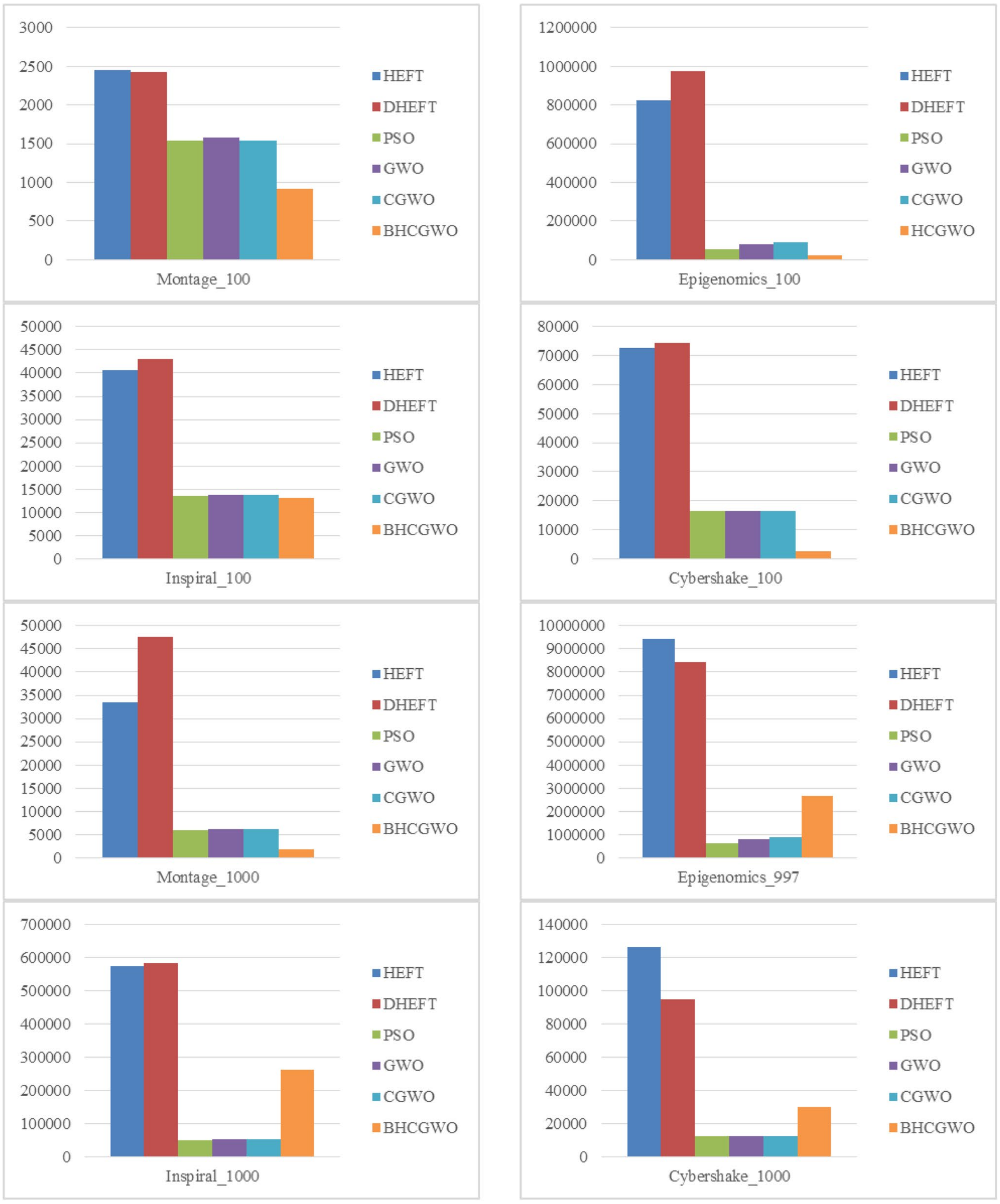

Fig. 17 Comparison of energy consumption

References

1. Rani D, Ranjan RK (2014) A comparative study of SaaS, PaaS and IaaS in cloud computing. Int J Adv Res Comput Sci Softw Eng 4(6):158–161
2. Masdari M et al (2016) Towards workflow scheduling in cloud computing: a comprehensive analysis. J Netw Comput Appl 66:64–82
3. Abualigah LMQ (2019) Feature selection and enhanced krill herd algorithm for text document clustering. Springer, Berlin
4. Aktel A et al (2017) The comparison of the metaheuristic algorithms performances on airport gate assignment problem. Transp Res Procedia 22:469–478
5. Gharehchopogh FS, Gholizadeh H (2019) A comprehensive survey: Whale optimization algorithm and its applications. Swarm Evol Comput 48:1–24
6. Shayanfar H, Gharehchopogh FS (2018) Farmland fertility: a new metaheuristic algorithm for solving continuous optimization problems. Appl Soft Comput 71:728–746
7. Gharehchopogh FS, Shayanfar H, Gholizadeh H (2019) A comprehensive survey on symbiotic organisms search algorithms. Artif Intell Rev 53:2265–2312
8. Mozaffari A, Emami M, Fathi A (2018) A comprehensive investigation into the performance, robustness, scalability and convergence of chaos-enhanced evolutionary algorithms with boundary constraints. Artif Intell Rev 52:2319–2380
9. Kennedy J (2011) Particle swarm optimization. In: Sammut C, Webb GI (eds) Encyclopedia of machine learning. Springer, Berlin, pp 760–766
10. Masdari M et al (2017) A survey of PSO-based scheduling algorithms in cloud computing. J Netw Syst Manag 25(1):122–158
11. Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845
12. Yang X-S, Hossein Gandomi A (2012) Bat algorithm: a novel approach for global engineering optimization. Eng Comput 29(5):464–483
13. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98
14. Abualigah L (2020) Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications. Neural Comput Appl 32:12381–12401
15. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249
16. Luo J, Chen M-R (2014) Improved shuffled frog leaping algorithm and its multi-phase model for multi-depot vehicle routing problem. Expert Syst Appl 41(5):2535–2545
17. Mirjalili SZ et al (2018) Grasshopper optimization algorithm for multi-objective optimization problems. Appl Intell 48(4):805–820
18. Kamboj VK (2016) A novel hybrid PSO–GWO approach for unit commitment problem. Neural Comput Appl 27(6):1643–1655
19. Kumar V, Kumar D (2017) An astrophysics-inspired Grey wolf algorithm for numerical optimization and its application to engineering design problems. Adv Eng Softw 112:231–254
20. Emary E, Zawbaa HM, Hassanien AE (2016) Binary grey wolf optimization approaches for feature selection. Neurocomputing 172:371–381
21. Kohli M, Arora S (2018) Chaotic grey wolf optimization algorithm for constrained optimization problems. J Comput Des Eng 5(4):458–472
22. Abdullah S, Alzaqebah M (2013) A hybrid self-adaptive bees algorithm for examination timetabling problems. Appl Soft Comput 13(8):3608–3620
23. Yousri D, Allam D, Eteiba M (2019) Chaotic whale optimizer variants for parameters estimation of the chaotic behavior in permanent magnet synchronous motor. Appl Soft Comput 74:479–503
24. Rizk-Allah RM, Hassanien AE, Bhattacharyya S (2018) Chaotic crow search algorithm for fractional optimization problems. Appl Soft Comput 71:1161–1175
25. Kumar Y, Singh PK (2018) A chaotic teaching learning based optimization algorithm for clustering problems. Appl Intell 49:1036–1062
26. Boushaki SI, Kamel N, Bendjeghaba O (2018) A new quantum chaotic cuckoo search algorithm for data clustering. Expert Syst Appl 96:358–372
27. Arora S, Anand P (2018) Chaotic grasshopper optimization algorithm for global optimization. Neural Comput Appl 31:4385–4405
28. Gandomi AH, Yang X-S (2014) Chaotic bat algorithm. J Comput Sci 5(2):224–232
29. Yu J, Buyya R (2005) A taxonomy of workflow management systems for grid computing. J Grid Comput 3(3–4):171–200
30. Etminani K, Naghibzadeh M (2007) A min–min max–min selective algorihtm for grid task scheduling. In: 2007 3rd IEEE/IFIP international conference in central asia on internet. 2007. IEEE
31. Gharehchopogh FS et al (2013) Analysis of scheduling algorithms in grid computing environment. Int J Innov Appl Stud 4(3):560–567
32. Topcuoglu H, Hariri S, Wu M-Y (1999) Task scheduling algorithms for heterogeneous processors. In: Proceedings. Eighth heterogeneous computing workshop (HCW'99). 1999. IEEE
33. Wei W, GuoSun Z (2007) Trusted dynamic level scheduling based on Bayes trust model. Sci China Ser F Inf Sci 50(3):456–469
34. Abdelkader DM, Omara F (2012) Dynamic task scheduling algorithm with load balancing for heterogeneous computing system. Egypt Inform J 13(2):135–145
35. Chen W, Deelman E (2012) Workflowsim: a toolkit for simulating scientific workflows in distributed environments. In: 2012 IEEE 8th international conference on E-science. 2012. IEEE
36. Rahman M, Venugopal S, Buyya R (2007) A dynamic critical path algorithm for scheduling scientific workflow applications on global grids. In: Third IEEE international conference on e-science and grid computing (e-science 2007). IEEE
37. Khajemohammadi H, Fanian A, Gulliver TA (2014) Efficient workflow scheduling for grid computing using a leveled multi-objective genetic algorithm. J Grid Comput 12(4):637–663
38. Fard HM et al (2012) A multi-objective approach for workflow scheduling in heterogeneous environments. In: 2012 12th IEEE/ACM international symposium on cluster, cloud and grid computing (ccgrid 2012). IEEE
39. Doğan A, Özgüner F (2005) Biobjective scheduling algorithms for execution time–reliability trade-off in heterogeneous computing systems. Comput J 48(3):300–314
40. Camelo M, Donoso Y, Castro H (2010) A multi-objective performance evaluation in grid task scheduling using evolutionary algorithms. Appl Math Inform 28:100–105
41. Durillo JJ, Prodan R (2014) Multi-objective workflow scheduling in Amazon EC2. Cluster Comput 17(2):169–189
42. Mateos C, Pacini E, Garino CG (2013) An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments. Adv Eng Softw 56:38–50
43. Selvarani S, Sadhasivam GS (2010) Improved cost-based algorithm for task scheduling in cloud computing. In: 2010 IEEE international conference on computational intelligence and computing research. IEEE
44. Mezmaz M et al (2011) A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. J Parallel Distrib Comput 71(11):1497–1508

45. Li J et al (2011) Cost-conscious scheduling for large graph processing in the cloud. In: 2011 IEEE international conference on high performance computing and communications. IEEE
46. Dongarra JJ et al (2007) Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems. In: Proceedings of the nineteenth annual ACM symposium on parallel algorithms and architectures. ACM
47. Sih GC, Lee EA (1993) A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. IEEE Trans Parallel Distrib Syst 4(2):175–187
48. Yu J, Kirley M, Buyya R (2007) Multi-objective planning for workflow execution on grids. In: Proceedings of the 8th IEEE/ACM international conference on grid computing. IEEE Computer Society
49. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength pareto evolutionary algorithm. TIK-report, 2001, 103
50. Deb K et al (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
51. Knowles J, Corne D (1999) The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: Congress on evolutionary computation (CEC99)
52. Filatovas E, Kurasova O, Sindhya K (2015) Synchronous R-NSGA-II: an extended preference-based evolutionary algorithm for multi-objective optimization. Informatica 26(1):33–50
53. Khalili A, Babamir SM (2017) Optimal scheduling workflows in cloud computing environment using Pareto-based Grey Wolf Optimizer. Concurr Comput Pract Exp 29(11):e4044
54. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
55. Burke EK, Bykov Y (2017) The late acceptance Hill–Climbing heuristic. Eur J Oper Res 258(1):70–78
56. Mukherjee A, Mukherjee V (2016) Chaotic krill herd algorithm for optimal reactive power dispatch considering FACTS devices. Appl Soft Comput 44:163–190
57. Saremi S, Mirjalili S, Lewis A (2014) Biogeography-based optimisation with chaos. Neural Comput Appl 25(5):1077–1097
58. Liang J-J, Suganthan PN, Deb K (2005) Novel composition test functions for numerical global optimization. In: Swarm intelligence symposium, 2005. SIS 2005. Proceedings 2005 IEEE
59. Yang X-S (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2(2):78–84
60. Mirjalili S, Lewis A (2013) S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evol Comput 9:1–14
61. Mahmoudi M, Gharehchopogh FS (2018) An improvement of shuffled frog leaping algorithm with a decision tree for feature selection in text document classification. 16(1):60–72
62. Masdari M, Zangakani M (2019) Efficient task and workflow scheduling in inter-cloud environments: challenges and opportunities. J Supercomput 76:499–535
63. Masdari M, Khoshnevis A (2019) A survey and classification of the workload forecasting methods in cloud computing. Cluster Comput 22:1–26
64. Xu Y et al (2014) A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Inf Sci 270:255–287
65. Schwiegelshohn U (2010) Job scheduling strategies for parallel processing. Springer, Berlin
66. Elsherbiny S et al (2018) An extended intelligent water drops algorithm for workflow scheduling in cloud computing environment. Egypt Inform J 19(1):33–55
67. Casas I et al (2018) GA-ETI: an enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments. J Comput Sci 26:318–331
68. Abazari F et al (2018) MOWS: multi-objective workflow scheduling in cloud computing based on heuristic algorithm. Simul Model Pract Theory 93:119–132
69. Alkhanak EN, Lee SP (2018) A hyper-heuristic cost optimisation approach for scientific workflow scheduling in cloud computing. Fut Gener Comput Syst 86:480–506
70. Choudhary A et al (2018) A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing. Fut Gener Comput Syst 83:14–26
71. Hu H et al (2018) Multi-objective scheduling for scientific workflow in multicloud environment. J Netw Comput Appl 114:108–122
72. Ebadifard F, Babamir SM (2018) Optimal workflow scheduling in cloud computing using AHP Based multi objective black hole algorithm. 2145:36–42
73. Yao G-S, Ding Y-S, Hao K-R (2017) Multi-objective workflow scheduling in cloud system based on cooperative multi-swarm optimization algorithm. J Cent South Univ 24(5):1050–1062
74. Fard HM et al (2012) A multi-objective approach for workflow scheduling in heterogeneous environments. In 2012 12th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid). IEEE
75. Naghibzadeh M (2016) Modeling and scheduling hybrid workflows of tasks and task interaction graphs on the cloud. Fut Gener Comput Syst 65:33–45
76. Masdari M, Zangakani M (2019) Green cloud computing using proactive virtual machine placement: challenges and issues. J Grid Comp. https://doi.org/10.1007/s10723-019-09489-9
77. Thaman J, Singh M (2017) Green cloud environment by using robust planning algorithm. Egypt Inform J 18(3):205–214