Cluster Computing (2019) 22:301–334
https://doi.org/10.1007/s10586-018-2856-x

Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in IaaS cloud computing environment

Syed Hamid Hussain Madni[1] · Muhammad Shafie Abd Latiff[1] · Shafi'i Muhammad Abdulhamid[2] · Javed Ali[3]

Received: 3 November 2017 / Revised: 11 June 2018 / Accepted: 14 October 2018 / Published online: 23 October 2018

Abstract
Resource scheduling is a procedure for the distribution of resources over time to perform a required task and a decision making process in cloud computing. Optimal resource scheduling is a great challenge and considered to be an NP-hard problem due to the fluctuating demand of cloud users and dynamic nature of resources. In this paper, we formulate a new hybrid gradient descent cuckoo search (HGDCS) algorithm based on gradient descent (GD) approach and cuckoo search (CS) algorithm for optimizing and resolving the problems related to resource scheduling in Infrastructure as a Service (IaaS) cloud computing. This work compares the makespan, throughput, load balancing and performance improvement rate of existing meta-heuristic algorithms with proposed HGDCS algorithm applicable for cloud computing. In comparison with existing meta-heuristic algorithms, proposed HGDCS algorithm performs well for almost in both cases (Case-I and Case-II) with all selected datasets and workload archives. HGDCS algorithm is comparatively and statistically more effective than ACO, ABC, GA, LCA, PSO, SA and original CS algorithms in term of problem solving ability in accordance with results obtained from simulation and statistical analysis.

Keywords Meta-heuristic algorithms · Resource scheduling · Cuckoo search · Gradient descent · Hybridization · Cloud computing

1 Introduction

Cloud computing is one of the most popular technologies that has become a fundamental part of the computing world nowadays. Its popularity and usage is growing every day and expected to increase further. Cloud computing provides IT services over the Internet in such a way that cloud user does not have knowledge about where the data or information is being stored, where the infrastructure is located and so on. The cloud users receive services without knowing any of the details about how it's provided [1, 2].

Organization spends time and money to scale up their IT infrastructure, such as hardware, software and services provisioning, to meet the business challenges. However, the scaling up process is slow with un-premises IT infrastructures. Cloud computing paradigm shift provides computing over the Internet. Cloud computing services contain highly optimized virtualized data centers providing hardware, software and information resources for use, whenever they are required [3]. In certain circumstances, cloud computing deals workload fluctuations and provides computation resources to manage large multimedia data and development environments. However, the success of cloud computing, resulting in more and larger data-centers,

✉ Syed Hamid Hussain Madni
madni4all@yahoo.com

Muhammad Shafie Abd Latiff
shafie@utm.my

Shafi'i Muhammad Abdulhamid
shafii.abdulhamid@futminna.edu.ng

Javed Ali
j.ali@seu.edu.sa

[1] Faculty of Computing, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia

[2] Department of Cyber Security Science, Federal University of Technology Minna, Minna, Niger State, Nigeria

[3] College of Computing Informatics, Saudi Electronic University, Madinah Munawarah, Kingdom of Saudi Arabia

creates new challenges at the level of infrastructure monitoring and management [4–6].

Resource scheduling is an important procedure in cloud computing that is used to take decision in resource distribution over time. However, there are some challenges in resource scheduling such as NP-hard problem due to fluctuating demand of cloud users and dynamic nature of various resources shown in Fig. 1. In cloud computing, the fitness function of resource scheduling is focused on cloud providers' and cloud users' objectives. Cloud providers want to enhance the utilization of resources by increasing the growth of revenue and profit, on the other side cloud users want to get maximum performance of required service with minimum cost or expenditure [7]. The following objective functions are generally considered for optimum resource scheduling in cloud computing including the availability, cost, energy, fault tolerance, load balancing, makespan, reliability, throughput, etc. An optimization problem is a function f mapping candidate solution to a fitness measure $= \boldsymbol{R^n} \rightarrow \boldsymbol{R}$. The optimization solution $z \in \boldsymbol{R^n}$ achieves the best optimal solution from all feasible solutions by proposed algorithm. Therefore it can be satisfies the minimization by Eq. (1).

$$f(x) < f(y), \forall (x, y) \in \boldsymbol{R^n} \tag{1}$$

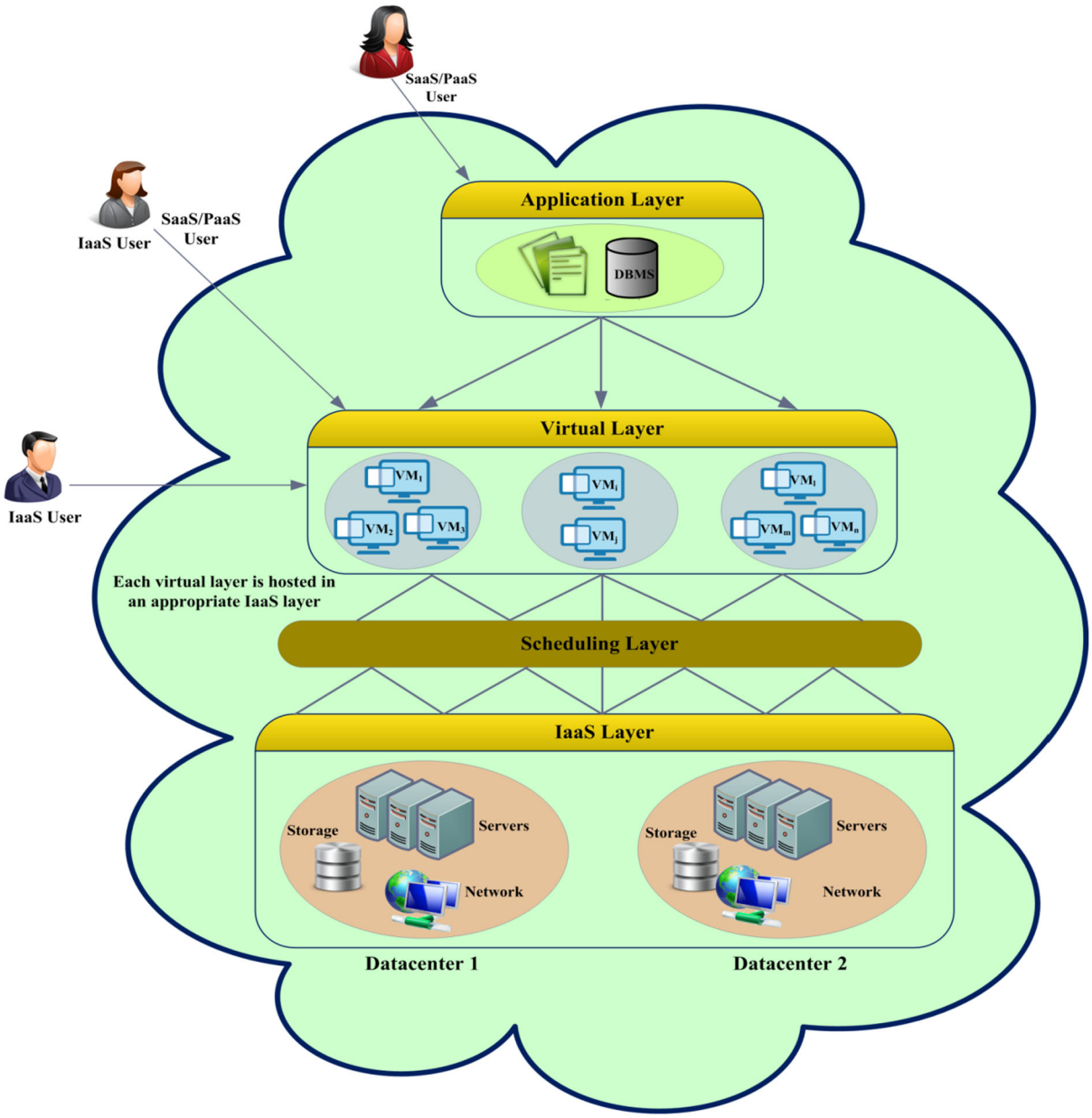

Fig. 1 Resource scheduling in cloud computing environment

Hybrid algorithm tries to attain all the factors whereas other algorithms fail to do so. Hybridization intentions gather the benefits of all algorithms in the form of a hybrid algorithm, even though at the same time try to reduce the extensive disadvantages. Generally, the consequences and outcomes of hybrid algorithms are commonly made some improvements in terms of either accuracy or speed. Optimal resource scheduling is considered to be a basic influence of cloud computing. It is only achieved by adopting the specific, enhanced and hybridization of greedy approaches with meta-heuristic algorithms.

CS algorithm is a nature-inspired meta-heuristic algorithm in which all entities have identical search behaviour. However, this simple uniform search behaviour is not always optimal to find the feasible solution for the specific problem and trap it into the local region leading to premature convergence. To overcome this weakness in this paper, we combined GD approach in term of local search with CS algorithm for enhancing the convergence rate, in order to assign a task to the specific Virtual Machine (VM) with lowest execution time to fulfill the cloud user's demand and enhance the resource utilization for cloud providers with less delay. In conclusion, a novel HGDCS algorithm with non-identical search strategy is proposed for resource scheduling in IaaS cloud computing, which reduces the makespan, throughput and degree of imbalance.

Our major contributions of this paper are as follows:

- Formulate the makespan, throughput and degree of imbalance through mathematical models for optimal resource scheduling as the objective functions.
- Hybridize the GD approach with CS algorithm for optimum resource scheduling in IaaS cloud computing.
- Design the HGDCS algorithm to address the proposed scheduling models.
- Implementation of the proposed HGDCS algorithm in CloudSim simulation tool.
- Performance evaluation of the existing meta-heuristic algorithms with HGDCS optimization algorithm by considering the matrices of makespan, throughput, degree of imbalance and performance improvement rate (%).
- Comparative and statistical analysis of the HGDCS algorithm with existing meta-heuristic algorithms for resource scheduling in IaaS cloud computing.

The remaining sections of this paper are systematically prepared as follows: In Sect. 2, we review the current comparison based studies and related works for resource scheduling algorithms in the area of IaaS cloud computing. Problem formulation is discussed in Sect. 3. We provide the description of local and global search, GD approach, CS algorithm and HGDCS algorithm in Sect. 4. Section 5 defines the parameters for evaluation, while Sect. 6 presents simulation setups for Case-I and Case-II for resource scheduling in cloud computing. Results and discussion show performance evaluation with the help of experimental simulation and statistical analysis in Sect. 7. The last section is Sect. 8, which consists of details of the conclusion, recommendation and future works.

2 Related works

In this section, we review the current comparison and existing studies of various heuristic, meta-heuristic and hybrid algorithms for resource scheduling in IaaS cloud computing. Tsai and Rodrigues [8] provide the systematic explanation of scheduling in cloud computing and also interrelated it with heuristic and meta-heuristic algorithms. Besides it, recommends instructions for the researchers to shift from heuristic algorithms to the meta-heuristic algorithms. In conclusion, meta-heuristic algorithms produce better solutions by modification of operators, alteration of the fitness function and hybridization with heuristic algorithms [9]. Further, Thaman and Singh [10] classify the algorithms as Greedy, Heuristics, Meta-heuristics and Genetic approaches based solutions for task scheduling in cloud computing.

The various meta-heuristic algorithms are analyzed for scheduling the cloud computing, including the ant colony optimization (ACO), genetic algorithm (GA), hill climbing, Particle swarm optimization (PSO), simulated annealing (SA) and Tabu Search with a discussion and future direction of a scheduling problem in cloud [8]. Further, Kalra and Singh [11] evaluate the ACO, BAT algorithm, GA, League Championship Algorithm (LCA) and PSO meta-heuristic algorithms for scheduling in grid and cloud computing with observations and open issue related to scheduling are also discussed. Similarly, Madni et al. [12] review and analyze the artificial bee colony (ABC), ACO, GA, immune algorithm (IA), LCA, PSO and SA meta-heuristic algorithms for assigning of resources in cloud computing. Also, provide the description of selected important parameters are applied for this problematic issue, including the cost, execution time, makespan, response time and utilization in cloud computing. In the same way, Hallaj and Tabbakh [13] discuss numerous swarm intelligence based algorithms for task scheduling and exactly emphasis on ABC algorithm. A systematic comparison is directed for presenting the features and uses of ABC for optimal scheduling in cloud computing. However, Huang and Ou [14] introduce and analyze the task scheduling algorithms including the ACO, GA and PSO algorithm in

cloud computing. Besides the good qualities each of them is slow in convergence speed. Also, Singh et al. [15] accomplish the an extensive review of various meta-heuristics-based techniques for scheduling in cloud computing including the ACO, bat algorithm (BA), GA, imperialists competitive algorithm (ICA), lion algorithm (LA), PSO and wind-driven optimization (WDO) algorithms with describing the various factors. Hence, these algorithms accomplish them no attention for resource scheduling in cloud computing environment.

Cui et al. [16] enhance the GA algorithm to accomplish a relevant task for the effective resource scheduling in cloud computing. Further, Chen et al. [17] merge GA algorithm with knapsack problem for utilization of resources and energy consumption in cloud computing. However, Sindhu and Mukherjee [18] combine the GA algorithm with various heuristic algorithms including the LCFP, SCFP and MCT for enhancing the convergence rate. Similarly, Javanmardi et al. [19] propose a hybrid job scheduling algorithm based on fuzzy theory and GA algorithm for decreasing the number of iteration to generate the population and accurate allocation of resources in order to node capacity. Moreover, Shojafar et al. [20] propose a hybrid algorithm based on fuzzy theory and GA (FUGA) algorithm for assigning the jobs to the optimal resources. For this purpose, fuzzy theory and GA modify with diverse fuzzy based stable state GA in order to minimize the makespan, execution cost, execution cost and average degree of imbalance. Later, Saha et al. [21] merge the GA algorithm and Queuing model as a tool for task scheduling algorithm for minimizing the waiting time and queue length for substantial the cloud users' demand in cloud computing.

In cloud computing, Zhang et al. [22] introduce the PSO algorithm in order to achieve the local and global search efficiently by weight inertia strategy and escape the struggling into a local optimum while Netjinda et al. [23] propose PSO algorithm for enhancing performance from the views of the entire cost and fitness convergence rate. The improvement in solution quality is perceived from the preliminary schedule with the variable neighbour search. Further, to minimize the consumption of energy and enhance the cloud providers' profit, job scheduling model is developed by using the PSO algorithm in cloud environment [24]. In addition, Abdi et al. [25] improve the PSO algorithm for enhancing the performance, merge with SJFP algorithm for improving the local search for reducing the makespan. However, Al-Olimat et al. [26] propose the hybrid scheduling algorithm by using the SA algorithm to enhance the performance of binary PSO algorithm. The proposed hybrid algorithm is used to reduce the makespan and enhance the utilization of resource in cloud computing.

For optimal resource scheduling, Wang and Yu [27] improve the Min-min algorithm for enhancing the performance of scheduling in cloud computing by considering the makespan and degree of imbalance. Hence, Li et al. [28] develop a Load Balancing aware ACO algorithm for task scheduling by considering the makespan and degree of imbalance in cloud computing. further, Tawfeek et al. [29] present a cloud task scheduling policy by using the ACO algorithm for reducing the makespan and degree of imbalance. While, Wen et al. [30] and Yang [31] combine the ACO and PSO for improving the local optimization for efficient resource scheduling in cloud computing. Similarly, Cho et al. [32] propose hybrid ACOPS algorithm for VM scheduling through load balancing by enhancing the convergence speed. In the same way, Liu et al. [33] introduce hybrid GA-ACO algorithm for optimal solution quickly by considering convergence speed for task scheduling. Optimized scheduling and VM validation are required for the optimum resource allocation in IaaS cloud computing. For this purpose, Muthulakshmi and Somasundaram [34] propose the Hybrid ABC-SA algorithm for efficient scheduling that improves the efficiency in terms of resource time searching by dynamic and random search.

Abdullahi, et al. [35] propose a discrete symbiotic organism search (DSOS) algorithm for an ultimate schedule of tasks to VMs in order to improve the convergence rate in cloud computing. Further, Abdullahi and Ngadi [36] merge SA and symbiotic organisms search (SOS) algorithms in SASOS algorithm for achieving ideal scheduling of tasks in order to attain better convergence ratio and quality of results in cloud computing. Similarly, Tsai et al. [37] improve the differential evaluation algorithm (DEA), by combining the Taguchi method for optimal resource scheduling. Improved DEA shows the ideal results for reducing cost and makespan. While, Guddeti and Buyya [38] propose hybrid bio inspired algorithm combination of modified PSO and cat swarm optimization (CSO) algorithms to assign a task to VMs in order to enhance the reliability, response time and resource utilization. Hence, Gabi et al. [39] put forward a new version of CSO for cloud task scheduling called Orthogonal Taguchi based-cat. The researchers explored the Taguchi approach to optimize task scheduling that returned minimum task makespan as compared to other existing tasks scheduling algorithms.

For optimized parameter mapping, Moon et al. [40] propose a slave ants based ACO (SACO) algorithm that schedules tasks to virtual machines (VMs) of cloud users in cloud computing environments in a competent way. The global optimization problem is solved by avoiding long paths with slave ants, whose pheromones are imperfectly accumulated by leading ants and minimal preprocessing. Experimental results SACO algorithm performed better than ACO algorithm for task scheduling while maximizing utilization of

Table 1 Summary of the related works for resource scheduling in IaaS cloud computing

References	Problems	Methods	Parameters	Achievements	Comparison methods
Huang and Ou [14]	Task scheduling	ACO, GA and PSO	Nil	Show the advantages of algorithms	Not implements
Cui et al. [16]	Resource scheduling	Improved GA	Execution time	Enhanced the performance	Min-min, max–min and GA
Chen et al. [17]	Resource scheduling	GA	Time	Verify the effectiveness of the algorithm	Not mentioned
Sindhu and Mukherjee [18]	Resource scheduling	GA-LCFP, GA-SCFP and GA-MCT	Makespan	Enhanced the performance	FCFS-RR
Javanmardi et al. [19]	Job scheduling	GA with fuzzy theory	Execution time, execution cost and degree of imbalance	Enhanced the performance	ACO
Shojafar et al. [20]	Job scheduling	Fuzzy theory and GA (FUGE)	Execution time, execution cost and degree of imbalance	Enhanced the performance	ACO
Saha et al. [21]	Resource scheduling	GA and queuing model	Waiting time	Enhanced the performance	FCFS
Zhang et al. [22]	Resource scheduling	PSO based hierarchical resource scheduling	Load balance and satisfaction	Avoid the local optima	PSO and best resource selection (BRS)
Netjinda et al. [23]	Cost optimization scheduling	PSO with neighbourhood search	Cost	Enhanced the performance	IaaS Cloud-partial critical paths (IS-PSP)
Liu et al. [24]	job scheduling	PSO	Energy and profit	Enhanced the performance	GA and random scheduling algorithm
Abdi et al. [25]	Resource scheduling	Modified PSO and shortest job fastest processor (SJFP)	Makespan	Avoid the local optima	GA and PSO
Al-Olimat et al. [26]	Resource scheduling	Hybrid PSO and SA	Makespan	Enhanced the performance	PSO
Wang and Yu [27]	Task scheduling	Improved Min-min	Execution time and degree of imbalance	Improved speed and quality of schedule	Min–min
Li et al. [28]	Task scheduling	Load balancing ACO	Makespan and degree of imbalance	Enhanced the performance	FCFS and ACO
Tawfeek et al. [29]	Task scheduling	ACO	Makespan and degree of imbalance	Enhanced the performance	FCFS and round robin (RR)
Wen et al. [30]	Resource scheduling	ACO-PSO	Execution time	Avoid the local optima	ACO
Yang [31]	Resource scheduling	ACO-PSO	Execution time	Avoid the local optima	ACO
Cho et al. [32]	Resource scheduling	ACO-PSO	Makespan and degree of imbalance	Enhanced the performance	ACO, PRACO, SA, GA and FCFS + RR
Liu et al. [33]	Task scheduling	GA-ACO	execution time	Avoid the local optima	ACO and GA
Muthulakshmi and Somasundaram [34]	Resource scheduling	ABC-SA	Makespan and response time	Enhanced the performance	Modified FCFS, shortest job first (SJF) and longest job first (LJF)
Abdullahi, et al. [35]	Task scheduling	Discrete symbiotic organism search (DSOS)	Makespan, response time and degree of imbalance	Improve the search space	PSO
Abdullahi and Ngadi [36]	Task scheduling	Simulated annealing based on symbiotic organism search (SASOS)	Makespan, response time and degree of imbalance	Improve the convergence rate and quality of solution	SOS

Table 1 continued

References	Problems	Methods	Parameters	Achievements	Comparison methods
Tsai et al. [37]	Resource allocation and scheduling	Improved DEA (IDEA)	Cost and time	Enhanced the performance	DEA and NSGA
Guddeti and Buyya [38]	Resource scheduling	Modified PSO and CSO (MPSO-CSO)	Time and resource utilization	Enhanced the Performance	ACO, MPSO, CSO, RR and exact
Gabi et al. [39]	Task scheduling	Orthogonal Tanguchi based cat swarm optimization (OTBCSO)	Makespan and degree of imbalance	Enhanced the performance	Min–Max, PSO-LDIW and HPSO-SA
Moon et al. [40]	Task scheduling	SACO	Makespan	Enhanced the performance	ACO and IACO
Gill et al. [41]	Resource scheduling	PSO BULLET	Execution cost, execution time and energy	Enhanced the users' satisfaction	PSO-HPC, PSO-SW and PSO-DVFS

cloud servers in cloud computing. Furthermore, Gill et al. [41] design the PSO based resource schedULing algorithm in cLoud EnvironmenT (PSO BULLET) for resource scheduling QoS constraints to execute the workloads on available resources in cloud computing efficiently. The simulations outcomes show that the proposed PSO BULLET technique competently decreases the execution cost, execution time and energy consumption along with other QoS parameters. Table 1 presents the summary of the related works for resource scheduling in IaaS cloud computing.

The analysis of the review revealed that the majority of the studies based on meta-heuristic algorithms such as ACO, ABC, GA and PSO. Usually, these algorithms have the drawback of trapping in local optima during scheduling in cloud computing. In addition, these proposed algorithms and techniques present more accurate results due to a comparison of basic or heuristics algorithms instead of other meta-heuristic algorithms. These comparisons are not sufficient for the true picture of algorithms performance. In contrast, this research is used eight (8) meta-heuristics algorithms to show the performance for near-optimal resource scheduling in IaaS cloud computing. Furthermore, proposed HGDCS algorithm avoids the local optima problem as well as performs better in case of global optima.

3 Problem formulation

Scheduling consists of the assignment of starting and completion times for the various operations to be performed. Like other scheduling problems, resource scheduling in cloud computing is a method that applies to the distribution of valuable cloud resources, generally processors, networks, storage and VMs to fulfill the demands of cloud users by the cloud providers. It is applied for balancing the load, ensure equal distribution of resources according to the demand and give some prioritization according to set rules. It also ensures that a cloud computing is able to serve all the cloud users' requests, with a certain quality of service.

Resource scheduling problem can clarify with the help of Eq. (2) and Fig. 1.

$$RS = \sum_{x=1}^{m,n} (R_x + S_x \ldots N_x) \times T_x \rightarrow U_x^Z \tag{2}$$

where it assigns m required numbers of cloudlets/task $T = (T_1, T_2, T_3, \ldots, T_m)$ onto n available physical resources to virtual resources in cloud data centers $R = (R_1, R_2, R_3, \ldots, R_n)$, $S = (S_1, S_2, S_3, \ldots, S_n)$ up to $N = (N_1, N_2, N_3, \ldots, N_n)$. The Fitness of particular objective $F = (F_1, F_2, F_3, \ldots, F_z)$ may be enhanced for the cloud users $U = (U_1, U_2, U_3, \ldots, U_n)$. When Z=1, the fitness function F_1 is assigned to the cloud users, when Z= 2, the fitness function F_2 is assigned to the cloud users and so on, according to the their demand.

Cloud computing consists of various datacenters and all datacenters are interrelated with VMs with different specification. Suppose there is a set of cloudlet/task $T_i = (T_1, T_2, T_3, \ldots, T_n)$ that are originated from the cloud users as their required demands. Cloud broker is responsible for assigning the cloudlet/task to requisite virtual resources $V_j = (V_1, V_2, V_3, \ldots, V_m)$ as virtual resources with minimum completion time. The expected time to completion (ETC) is described as the expected time of all cloudlets/tasks are execute on a definite virtual resource acquired by using ETC matrix as shown in Eq. (3). Total number of cloudlets/tasks multiply by the total number of resources gives the ETC matrix's dimension and their elements are characterized as an ETC (T_i, V_j).

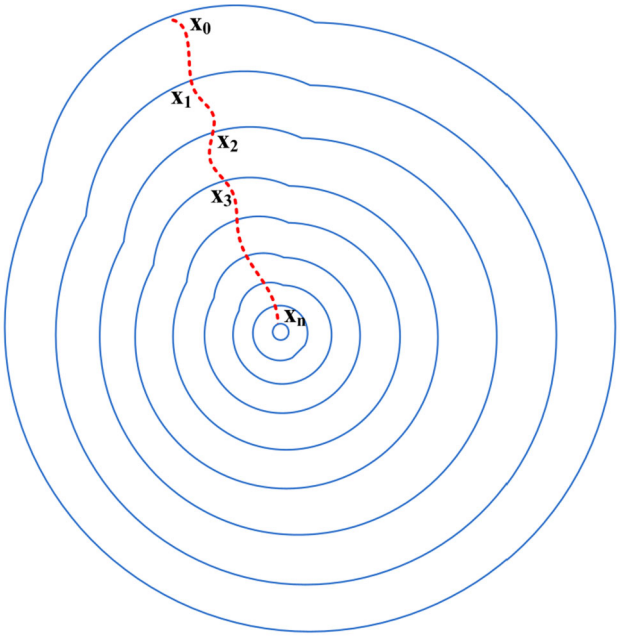

Fig. 2 Behaviour of gradient descent method

$$ETC(T_i, V_j) = \begin{bmatrix} T_1V_1 & T_1V_2 & T_1V_3 & \dots & \dots & T_1V_m \\ T_2V_1 & T_2V_2 & T_2V_3 & \dots & \dots & T_2V_m \\ T_3V_1 & T_3V_2 & T_3V_3 & \dots & \dots & T_3V_m \\ \vdots & \vdots & \vdots & \dots & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \dots & \vdots \\ T_nV_1 & T_nV_2 & T_nV_{i3} & \dots & \dots & T_nV_m \end{bmatrix} \quad (3)$$

Therefore, the main objective of this research paper is to hybridize the Gradient decent approach onto local search of optimized CS algorithm for mapping the cloudlets/tasks on virtual resources with minimum ETC in order to minimum makespan and throughput with a balanced degree of imbalance by enhancing the convergence rate.

3.1 Mathematical models for the resource scheduling problem

This study considers the makespan, throughput and degree of imbalance as the objective functions for optimal resource scheduling in cloud computing. Therefore, the fitness value of a HGDCS can be calculated for makespan, throughput and degree of imbalance by using the Eqs. (4), (5) and (6), respectively.

3.1.1 Makespan model

$$f(x) = \max \bigcup_{i=1}^{m} C_i, \quad \forall\, i \,\epsilon\, N, \quad i = 1, 2, 3, \dots\dots, m \quad (4)$$

3.1.2 Throughput model

$$f(x) = \sum_{i=1}^{m} C_i, \quad \forall i \epsilon N, \quad i = 1, 2, 3, \dots\dots, m \quad (5)$$

3.1.3 Degree of imbalance model

$$f(x) = \frac{\bigcup_{i-1}^{m} \max C_i - \bigcup_{i-1}^{m} \min C_i}{\text{avg } C_i}, \quad \forall i \epsilon N, \quad i = 1, 2, 3, \dots\dots, m \quad (6)$$

where C_i indicates the completion time of specific cloudlet/task. The lesser makespan and throughput, while smooth and stable degree of imbalance show the better efficiency of the proposed algorithm. Our main objective is to reduce the completion time of specific cloudlet/task on

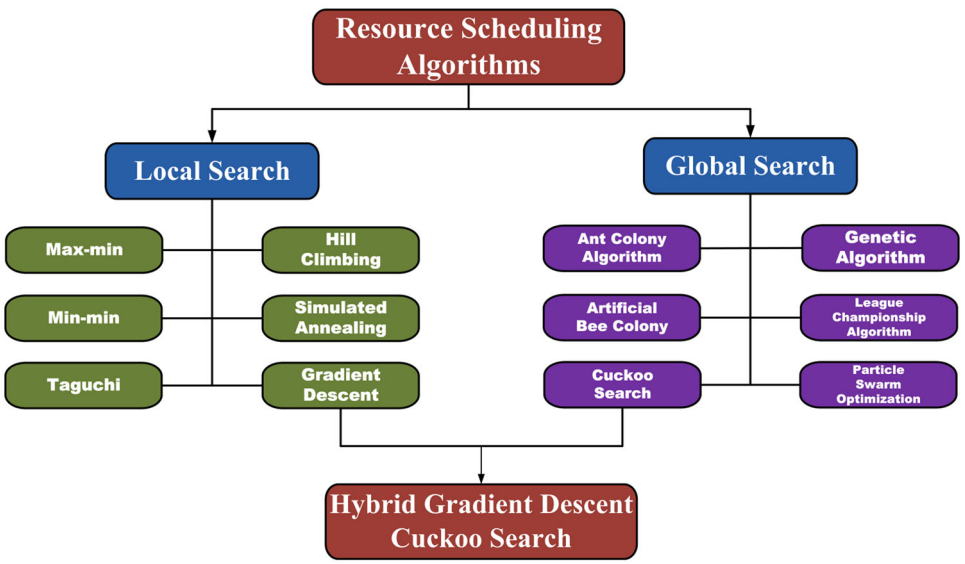

Fig. 3 Composition of hybrid gradient descent cuckoo search (HGDCS) algorithm

Pseudo-Code of Hybrid Gradient Descent Cuckoo Search Algorithm

Input: (P_α, λ).
Output: S_{best}

```
f(x), x = (x_1, ..., x_d)^T;                 // Objective function
Initial x_i (i = 1, 2, ..., n),              // population of n host nests
P_α ∈ [0,1] and Max_itr;
while : ((t < Max_itr) or (Stop Condition)) do
        Get a cuckoo (say i) randomly by Eq 8 and Eq 9 or Eq 11;
                                             // new solution x_i^(t+1)
        Check F_i = f(x_i^(t+1))             // evaluate its quality/fitness F_i
        Choose a nest among n (say j) randomly; // old solution x_i^t
        if (F_i > F_j) then                  // x_i^(t+1) > x_i^t
            F_j ← F_i ;                      // Replace old solution x_i^t with new solution x_i^(t+1)
        end if
        if (rand [0,1] < P_α) then           // abandon a fraction (P_α) of worse nests and
            init_nest (x_worst);             // build new ones at new locations
        end if
        if (F_i < F_min)then                 // rank the solutions and find the current best
            x_best = x_i;
            F_i = F_min;                     // keep the best solutions or nest with quality solutions
        end if
        t←t+1;
end while
return S_best ;
end
```

Fig. 4 Pseudo-code of hybrid gradient descent cuckoo search (HGDCS) algorithm

all VMs during the resource scheduling, then we get minimum makespan, throughput and stable degree of imbalance.

4 Methodology

This section presents the description of the local and global search, the basic structure of GD approach, standard CS algorithm and proposed HGDCS algorithm in detail.

4.1 Local search versus global search

A search technique which always reaches the same locally optimal solution from the same starting point is probably a local search technique. Equally, the performance of a global search technique should be less dependent on its initial position(s). Whereas a local search technique will target nearby local optima, global search techniques should be able to find (local) optima anywhere in the search space.

Both search techniques attempt to find a solution that optimizes a cost criterion. Local search algorithms start exploring the state space in a certain point of the state space (this point can be selected using a huge variety of techniques. These techniques are highly dependent on the problem domain and the local search algorithm), and iteratively try to find a better solution in terms of the cost function. In general, these algorithms are faster than other global search techniques and they can provide quite good solutions if the initialization step is adequated to the problem. Also, these algorithms are iterative, and we always know the best found solution at the current iteration. This leaves us total freedom to select the stop condition. As other answers point, these algorithms only provide local optimas, which may have a much higher cost than the global one, and which also depend on the initial solution in which the exploration started.

Ideally speaking, a global searching technique is promised to make sure to find the best global formation but this is achieved mostly at the cost of a long time searching. But then again in reality, they are run and stop when the stopping criterion comes across. Examples of this search include the PSO, SA, GA, etc. Whereas, local search algorithms do not totally focus on search and but it attempts to move from a current formation to a neighboring refining formation. This is much depending on the initial search space and initial formation. An example of a local search is hill climbing algorithm, which is an iterative algorithm which can start with a random solution and then after the algorithm tries to find a better solution by incrementally altering a solution of a single element. If this alteration harvests a better solution, an incremental alteration is made to a new solution, this process can be repeated in anticipation of no more enhancement is identified.

There are NP problems where finding one optimal and the definite solution are not possible. From a classification

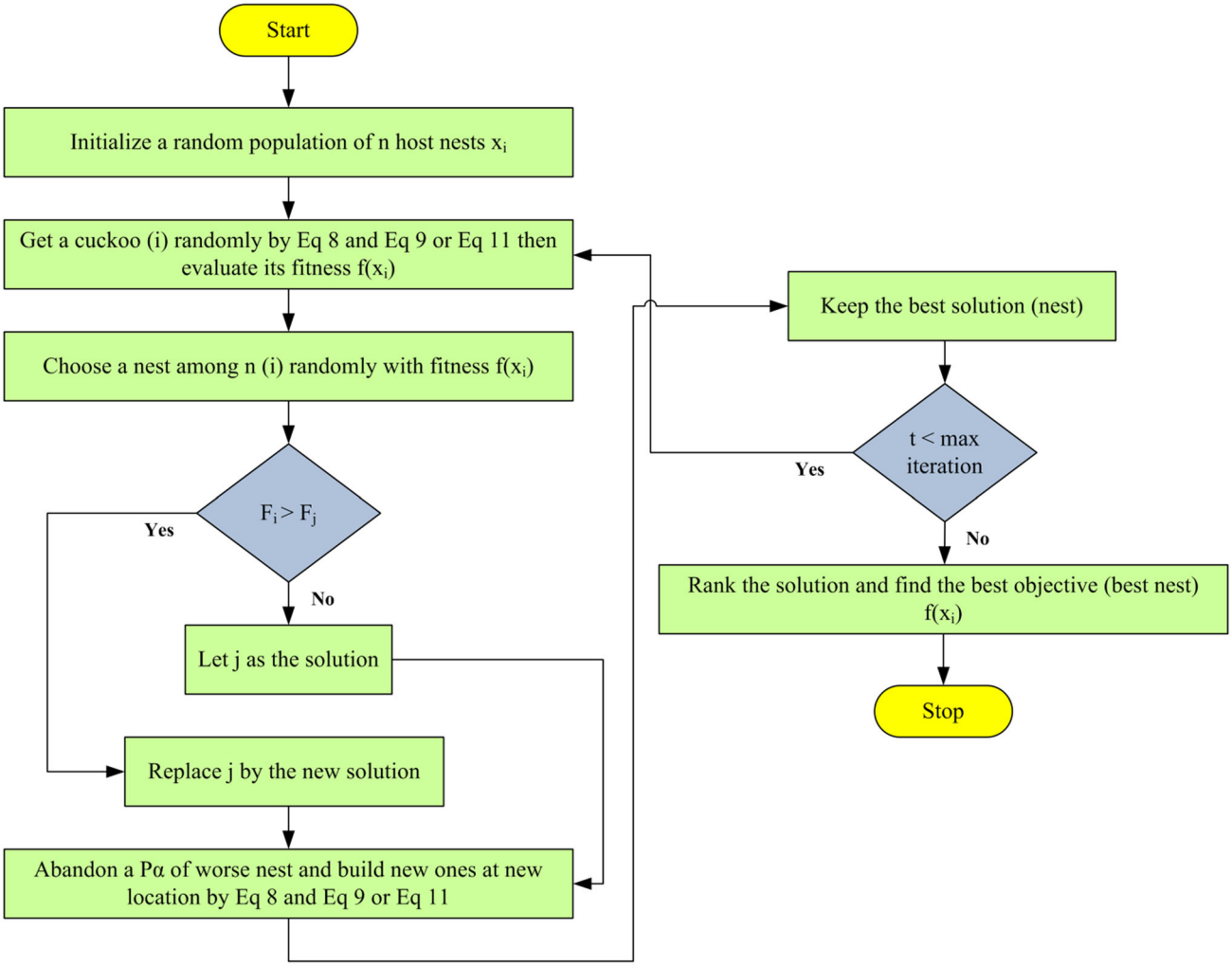

Fig. 5 Flow chart of hybrid gradient descent cuckoo search (HGDCS) algorithm

point of view, this differs from the method used to find the solution. Any method that searches in the vicinity of a starting point (incremental methods) and has the potential to get stuck the moment it sees extrema is a local method. GD approach is the classic example for this. Global methods will usually treat the whole feature space as one when finding the best solution. A very slow and a primitive example would be the exhaustive search.

4.2 Gradient descent approach

Gradient descent (GD) is a principal order iterative optimization method. It is used for finding a local minimum of a function, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. If instead one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function, the procedure is then known as a "Sleepest Ascent Method" [42, 43].

Gradient descent is founded on the observation that if the multi-variable function **F(x)** is distinct and differentiable in a neighborhood of a point **a**, then **F(x)** decreases fastest if one goes from **a** in the direction of the negative gradient of **F** at **a**, $-\nabla \boldsymbol{F}(\boldsymbol{a})$. It follows that, if

$$b = a - \gamma \nabla F(a) \tag{7}$$

where a is the current position, b is the next position, γ is the weight factor, $\nabla F(a)$ is the direction of sleepest ascent.

For $\boldsymbol{\gamma}$ minor sufficient, then $\boldsymbol{F}(\boldsymbol{a}) \geq \boldsymbol{F}(\boldsymbol{b})$. In other words, the term $\boldsymbol{\gamma}\nabla \boldsymbol{F}(\boldsymbol{a})$ is subtracted from **a** because we want to move against the gradient, namely down toward the minimum. With this observation in mind, one starts with a guess x_0 for a local minimum of **F,** and considers the sequence $\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2} \ldots$ such that:

$$x_{n+1} = x_n - \gamma_n \nabla F(x_n), n > 1 \tag{8}$$

then

$$F(y_i) = \alpha(x_i - y_i)^2 + \beta(y_i - y_{i+1})^2 + \beta(y_i - y_{i-1})^2 \quad (9)$$

or

$$y_i^{'} = y_i + 2\alpha(x_i - y_i) + 2\beta(y_{i+1} - y_i - 2y_i)^2 \quad (10)$$

So we have x_0

$F(x_0) \geq F(x_1) \geq F(x_2) \ldots$.

So hopefully the sequence $\boldsymbol{x_n}$ converges to the desired local minimum. Note that the value of the step size $\boldsymbol{\gamma}$ is allowed to change with each iteration. With certain assumptions on the function **F** and particular choices of $\boldsymbol{\gamma}$, convergence to a local minimum can be guaranteed.

The behaviour of GD approach is illustrated in Fig. 2. Here **F** is assumed to be defined on the plane, and that its graph has oval shapes. The blue curves lines are the contour lines, that is, the regions on which the value of **F** is constant. A red dotted line creating at a point shows the direction of the negative gradient at that point $\mathbf{x_0}$ to $\mathbf{x_n}$. Note that the (negative) gradient at a point is orthogonal to the contour line going through that point. We see that gradient descent indicates to the center of the oval shape, that is, the point where the value of the function **F** is minimal.

Table 2 Parameters setting of meta-heuristic algorithms in cloud computing for resource scheduling

Algorithms	Parameters	Values
ACO	Number of ants	10
	Vaporization factor, ρ	0.4
	Pheromone tracking weight, α	0.3
	Heuristic information weight, β	1
	Pheromone updating constant, Q	100
ABC	Number of scout bees	1000
	Number of sites	5
	Number of best sites	1
	Number of bees for e sites	800
	Number of bees for other e-m sites	200
CS	Population size	20
	Abandon probability $P\alpha$	0.25
	Max iteration	1000
	Step size λ	0.01, 1
GA	Population size	1000
	Max iteration	1000
	Crossover rate	0.5
	Mutation rate	0.1
PSO	Particle size	100
	Self-recognition coefficients, $c1$, $c2$	2
	Uniform random number, $R1$	0,1
	Max iteration	1000
	Inertia weight, W	0.9–0.4
SA	Initial temperature, F_{init}	10
	Final temperature, F_{final}	0.001
	Cooling rate, δ	0.9

4.3 Cuckoo search algorithm

Cuckoo search (CS) is created by "Xin-She Yang" and "Suash Deb". CS has been originated to contribute in global optimization problems [44]. Cuckoo in real is a captivating bird, not only because of the sweet sound but also due to the aggressive nature of reproduction. Some of the species of cuckoo bird "Ani" and "Guira" lay their eggs in the commune's nest. Although, they may pass through other bird's eggs to maximize the hatching probability of their own [45].

(1) Cuckoo's eggs, nest or cuckoo represents as a solution of the algorithm.
(2) Flight from one nest to another nest is also a solution for updating or modification.
(3) If the host bird identifies the cuckoo's eggs in the nest is considered as the worst solution, it will not be fruitful for further processing.

Assumptions of CS algorithm

(1) Cuckoo bird lays one egg at a time, which shows one solution.
(2) Nests having the best solution would be the finest nests, that means nests itself represent the solution.

Table 3 Simulation parameters setting of CloudSim for Case-I in cloud computing for resource scheduling

Sr. No	Entities	Parameters	Values
1	User	Number of users	50
		Number of brokers	5
2	Cloudlet	Number of cloudlets	100–1000
		Length	800,000
		File size	600
3	Host	RAM	2048 MB
		Storage	1,000,000
		Bandwidth	10000
4	VM	No of VMs	25
		Type of policy	Time shared
		RAM	512 MB
		Bandwidth	10000
		MIPS	1000
		Size	10000
		VMM	Xen
		Operating system	Linux
		Number of CPUs	1 on each
5	Data center	Number of data centers	2

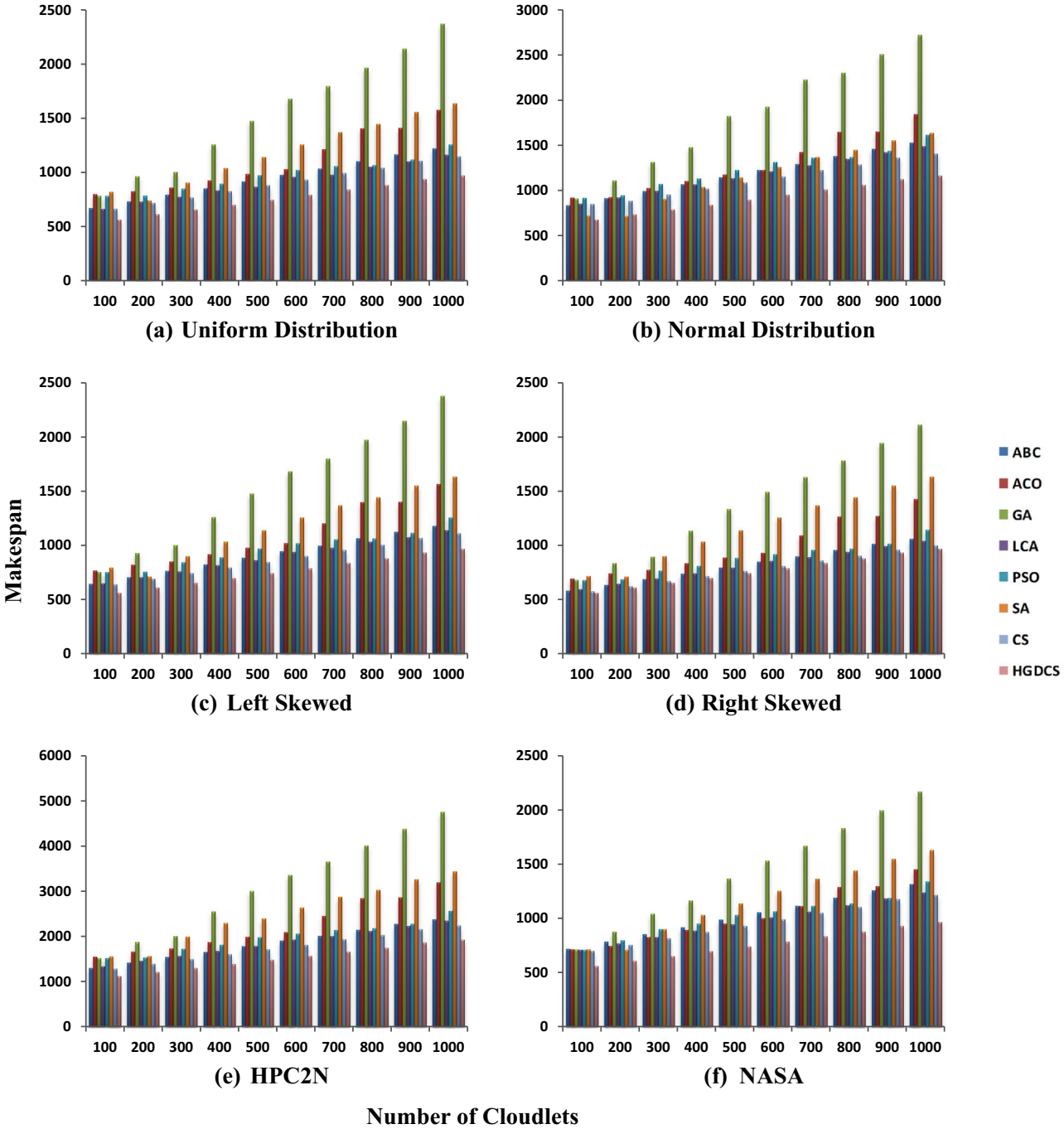

Fig. 6 Makespan for resource scheduling in Case-I using **a** uniform distribution, **b** normal distribution, **c** left skewed, **d** right skewed **e** HPC2N and **f** NASA

(3) A fixed number of the nests would be available; there are finite numbers of initial solutions, which will remain same throughout the algorithm.

Pα depicts the probability of finding alien's eggs that is represented as $P\alpha \in [0, 1]$. Lowering Pα results in reducing the chances of finding alien's eggs by the host bird and vice versa.

Three parameters are used in CS algorithm

- $P\alpha \in [0, 1]$ where the probability of worse nest is to be abandoned
- $\alpha > 0$

The step size α that is related to the scale the problem in most cases is greater than 1 i.e. $\alpha > 1$.

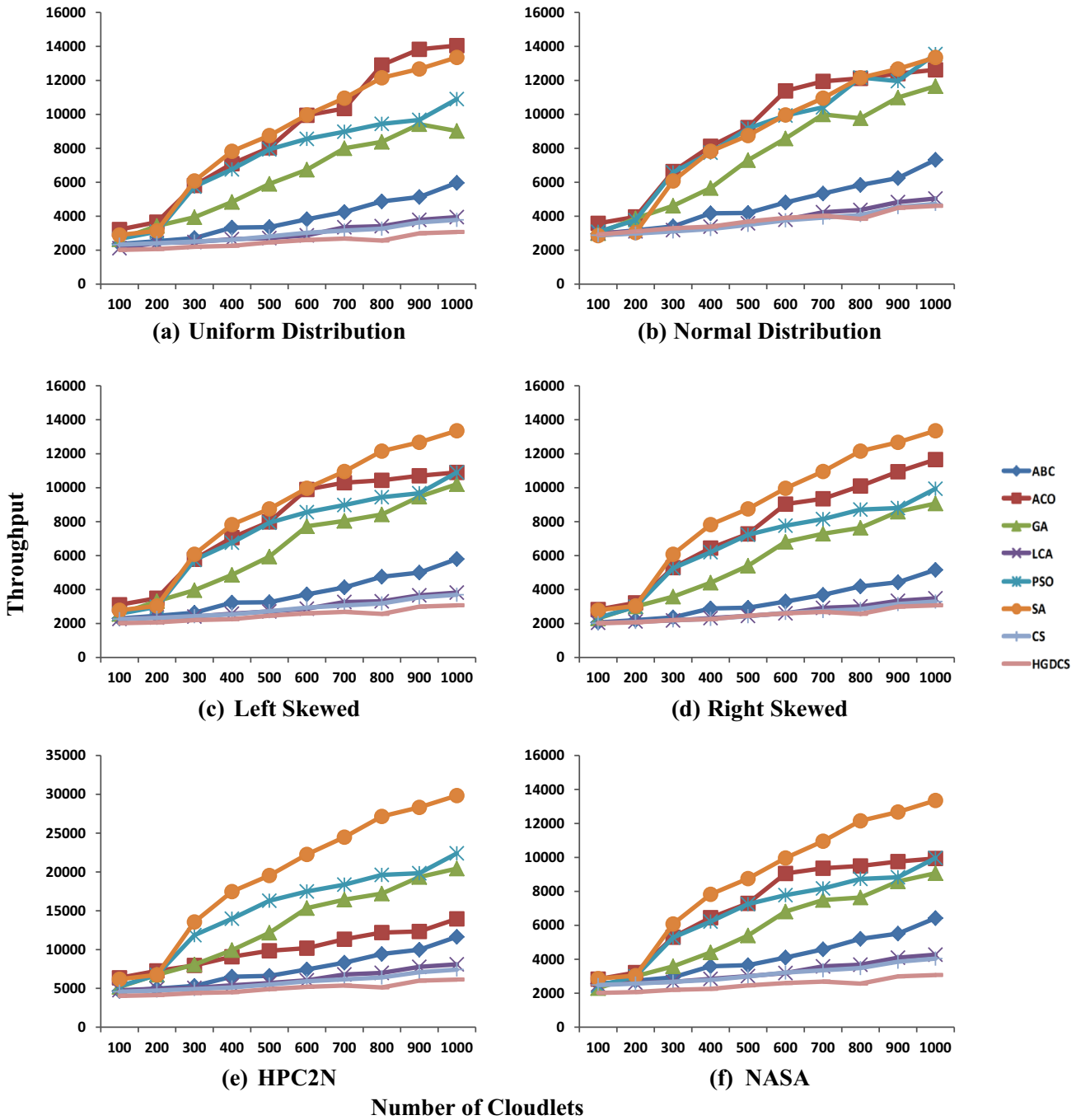

Fig. 7 Throughput for resource scheduling in Case-I using **a** uniform distribution, **b** normal distribution, **c** left skewed, **d** right skewed **e** HPC2N and **f** NASA

- λ step length random.

4.3.1 Significance of cuckoo search algorithm

CS has essentially two searching abilities that include the local and global search, which is controlled with the help of a discovery and switching probability. Discussed before, the local search is very insensitive with about ¼ of the search time (for $P\alpha=0.25$), on the other hand, the global search takes about of the search time (for $1-P\alpha=0.75$). This shows that the search space is explored more efficiently on the global scale, with higher probability.

CS algorithm is considered as an effective technique due to the less number of iterations and convergence rate. The decreased input data is the cause of its smaller number of

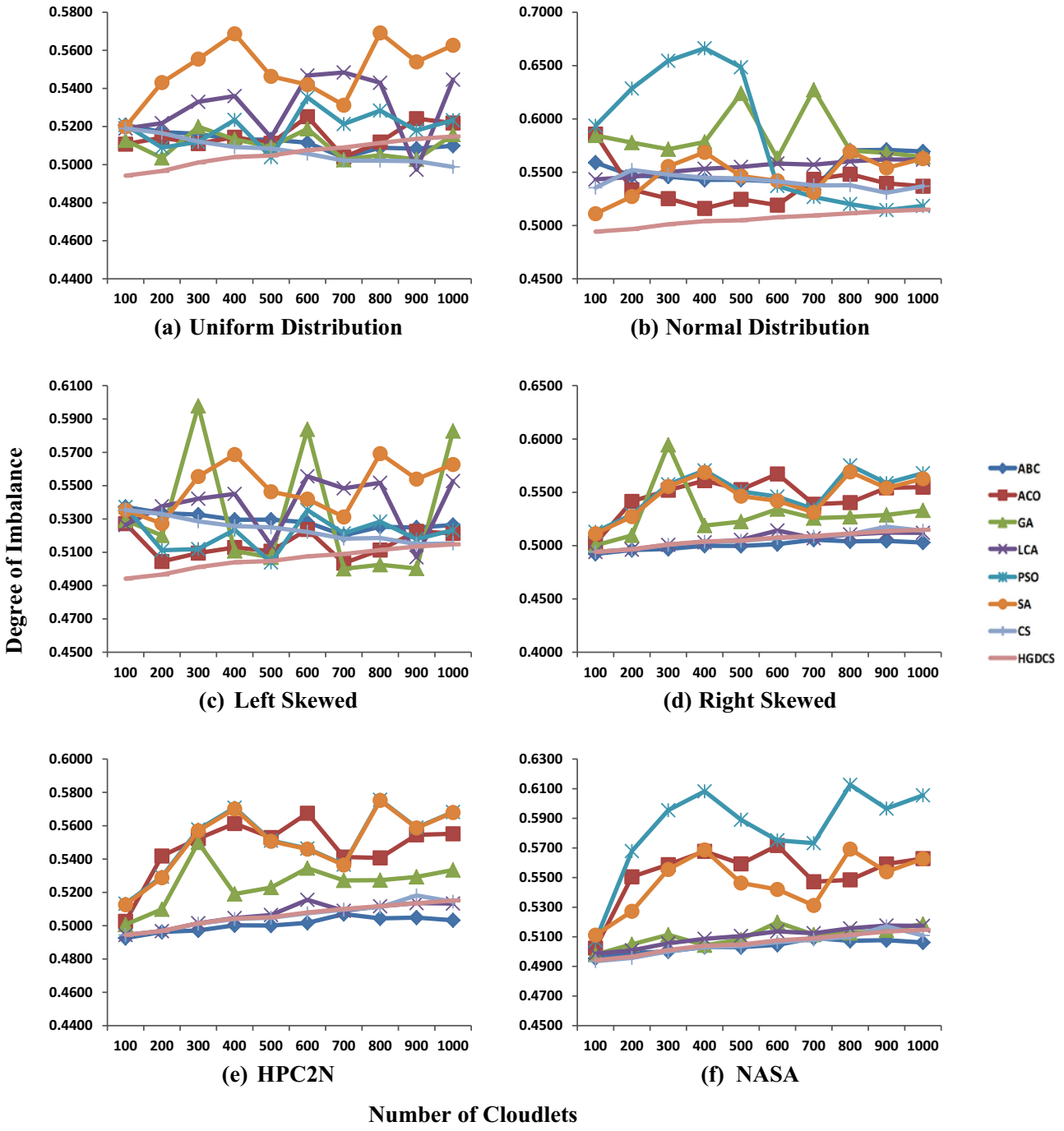

Fig. 8 Degree of imbalance for resource scheduling in Case-I using **a** uniform distribution, **b** normal distribution, **c** left skewed, **d** right skewed **e** HPC2N and **f** NASA

calculations and input data. By adding Levy flight function, the global search is conducted simultaneously as local search in CS algorithm. Due to these reasons, it is extensively rewarded attention by researchers to get the optimal results in multi-discipline areas [46–50].

The global search of CS uses the Levy flights rather random walk strategy. Due to the fact theta, the Levy flights have infinite mean and variances. Therefore, CS explores more efficiently than other algorithms which depend on standard Gaussian processes. This benefit combines with both searches capabilities (local and global), which provides the guarantee of global convergence [51–54].

Table 4 Statistical significance of meta-heuristic algorithms for makespan in Case-I after 50 runs

Algorithm	ACO			ABC			GA			LCA		
Statistical dispersion/ number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
100	795.84	2.04	793.66	667.53	0.58	667.10	781.01	3.55	774.06	659.89	1.16	657.57
500	982.61	6.59	974.90	914.39	0.69	913.37	1472.32	7.34	1434.73	863.53	2.29	863.17
1000	1575.77	9.14	1560.91	1218.15	13.41	1191.80	2370.83	4.40	2365.51	1159.24	1.53	1158.37
S02												
100	918.03	2.23	914.91	832.09	5.55	831.14	906.29	3.47	899.80	849.88	5.06	848.04
500	1174.76	4.18	1170.13	1143.32	1.08	1141.67	1821.21	91.77	1775.03	1133.08	5.37	1132.64
1000	1842.63	17.13	1813.71	1525.33	19.88	1486.15	2719.24	5.46	2709.52	1486.66	0.71	1485.48
S03												
100	770.97	1.95	768.74	647.11	1.64	646.59	756.69	3.31	750.26	650.63	1.40	648.20
500	977.60	6.37	970.16	885.96	1.76	884.81	1479.44	72.72	1441.68	863.53	1.29	863.17
1000	1567.70	8.94	1553.65	1180.00	12.60	1155.26	2380.45	4.42	2375.07	1142.19	0.54	1141.29
S04												
100	693.39	1.86	691.69	580.78	1.40	580.58	680.28	3.36	673.67	596.27	1.89	594.57
500	888.62	9.60	877.54	796.40	2.50	795.80	1336.34	66.24	1302.30	793.76	2.30	793.76
1000	1425.72	15.00	1399.15	1061.46	12.41	1037.04	2115.71	3.92	2110.11	1042.68	0.47	1042.00
S05												
100	1562.73	4.18	1558.90	1308.93	2.91	1308.48	1533.18	7.57	1518.29	1351.64	2.09	1347.63
500	2002.73	21.64	1977.76	1794.87	2.12	1793.52	3011.76	149.28	2935.06	1797.63	2.33	1797.63
1000	3213.22	33.81	3153.32	2392.26	27.97	2337.22	4768.28	8.84	4755.65	2355.25	1.11	2353.64
S06												
100	719.02	1.92	717.26	722.96	2.49	722.68	718.26	3.53	711.33	713.17	2.08	711.11
500	954.25	10.88	941.73	991.45	1.61	990.69	1374.29	67.96	1339.27	948.69	3.03	948.66
1000	1457.44	16.85	1428.03	1321.50	15.53	1290.92	2174.43	4.05	2168.42	1253.13	3.58	1242.27

Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
100	779.32	3.96	688.70	818.32	9.96	727.70	660.83	1.31	658.37	620.98	1.28	618.76
500	973.42	5.70	967.00	1140.98	9.06	1130.25	876.44	0.81	875.30	818.01	0.78	816.94
1000	1256.23	3.25	1251.50	1634.08	11.25	1622.31	1146.73	0.47	1145.77	1062.71	0.46	1061.76
S02												
100	915.06	60.54	777.80	717.20	37.31	632.59	846.58	4.62	816.91	677.98	1.33	674.96
500	1221.96	17.87	1201.08	1140.98	9.07	1130.25	1083.43	1.05	1082.17	894.86	0.84	893.54
1000	1614.99	4.36	1608.35	1634.09	11.25	1622.31	1409.23	0.65	1407.97	1164.72	0.48	1163.76
S03												
100	755.39	5.37	670.64	794.39	7.33	709.64	640.53	1.31	638.35	615.11	1.26	612.86
500	973.42	5.71	967.00	1140.98	9.07	1130.25	848.92	0.80	847.86	810.42	0.77	809.36
1000	1256.23	3.25	1251.50	1634.09	11.25	1622.31	1109.98	0.48	1109.02	1053.05	0.45	1052.13
S04												
100	678.20	7.31	593.59	717.20	3.31	632.59	575.09	1.11	572.57	565.08	1.11	562.56
500	884.48	7.56	875.54	1140.98	9.07	1130.25	763.86	0.71	762.76	745.85	0.71	744.74
1000	1145.76	8.44	1136.94	1634.09	11.25	1622.31	1000.80	0.40	1000.00	970.76	0.40	959.96
S05												
100	1528.49	8.09	1337.81	1567.13	8.02	1376.62	1296.12	2.51	1290.43	1129.60	2.22	1124.56
500	1993.40	17.03	1973.25	2405.96	20.42	2381.80	1721.55	1.60	1719.07	1490.94	1.41	1488.74

Table 4 continued

Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
1000	2582.26	19.02	2562.36	3450.63	25.34	3424.12	2255.55	0.91	2253.74	1940.56	0.80	1938.96
S06												
100	711.97	8.55	624.55	717.20	7.31	632.59	704.35	1.36	701.27	565.08	1.11	562.56
500	1036.11	11.25	1022.80	1140.98	9.07	1130.25	935.51	0.87	934.17	745.84	0.70	744.74
1000	1346.02	16.05	1327.28	1634.09	11.25	1622.31	1219.35	0.49	1218.37	970.76	0.40	969.96

S01 uniform distribution, S02 normal distribution, S03 left skewed, S04 right skewed, S05 HPC2N, S06 NASA)

4.4 Hybrid gradient decent cuckoo search (HGDCS) algorithm

In the existing meta-heuristic algorithms proposed the solutions for optimization issues of resource scheduling in IaaS cloud computing, but the consequences and outcomes show some limitations such as less accuracy and speed. One algorithm is effective but not all the time only might be effective to resolve the optimization problems and change the behaviour due to the large amount of heavy load, which indirectly effects on the performance and objective function. There are many optimization algorithms are proposed for resource scheduling in the literature of cloud computing, but no single algorithm is suitable for all problems.

To overcome these drawbacks, HGDCS algorithm is implemented in IaaS cloud computing for resource scheduling, fast local optimization achieves due to the GD approach and global optimization maintains through the breeding and behaviour skills of the optimized CS algorithm shown in Fig. 3.

HGDCS algorithm uses the sensible arrangement of local and global search and controls them by switching parameter Pα. The local search is calculated by using GD approach with the help of Eqs. (8) and (9). However, the global search is calculated using Levy flight with the help of Eq. (11).

Levy flight is a random walk in which the steps are defined in terms of the step length, which are distributed according to a heavy tailed probability distribution with the direction of steps being isotropic and random.

$$x_i^t = x_i^{t+1} + \alpha \oplus Levy(\lambda) \tag{11}$$

where x_i^t is the new solution, x_i^{t+1} is the current solution and $\alpha \oplus \text{levy}(\lambda)$ is thetransaction probability

For the better understanding of HGDCS algorithm, the pseudo-code and flow of the HGDCS algorithm are shown in Figs. 4 and 5. The first step is initialization, in which the fixed numbers of cuckoo's eggs are generated randomly. Fitness represents a numbers of inter-partition connections. The low value of inter-partition is better for the quality of the solution. After finding the fitness values of all eggs, select the best having the least value of inter-partition cuts. All cuckoo fly towards the best nest, this represents the all random generated moves toward the best solution, with step-size variable decides how is the distance traveled by a cuckoo towards the best in one iteration. Selection of n best solution from $2n-1$, total solution, where n solution is generated from previous solution and $n-1$ solution from the new generation. With the help of iteration criteria, achieve the better quality of the solution. There are two criteria to stop the iteration either achieving the required quality of the solution or by fixing the number of iteration. Both criteria are depending upon the application of cuckoo search algorithm.

5 Performance metrics

Resource scheduling parameters are used for evaluating the scheduling algorithms in cloud computing environment. These parameters are based on computing, network and storage including the availability, bandwidth/speed, cost, degree of imbalance, energy, execution time, memory, performance, priority, reliability, response time, SLA, temperature, throughput, time and utilization [7, 55, 56]. This study considers the makespan, throughput, degree of imbalance and performance improvement rate. These performance metrics are discussed below:

5.1 Makespan

Makespan uses to determine the maximum completion time, by calculating the finishing time of the latest task, when all tasks are scheduled. If the makespan of specific cloudlet or task is not minimized then the demand will not be completed on time [57, 58]. This study reduces the

Table 5 Statistical significance of meta-heuristic algorithms for throughput in Case-I after 50 runs

Algorithm	ACO			ABC			GA			LCA		
Statistical dispersion/ number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
100	3217.22	20.09	2922.59	2367.63	27.44	2344.57	2640.35	31.54	2615.25	2117.97	19.13	2072.74
500	8031.71	130.67	7987.74	3359.59	199.23	3113.48	5908.08	428.30	5498.96	2709.76	72.46	2596.41
1000	14,045.46	102.11	13,888.32	5968.79	998.48	5068.21	9021.12	289.49	8932.56	3949.76	224.61	3602.17
S02												
100	3582.43	43.03	3247.13	2965.19	36.45	2932.78	3001.86	25.53	2977.63	2998.70	25.44	2938.54
500	9219.69	94.21	9101.91	4193.03	239.03	3912.96	7300.61	572.33	6702.05	3582.21	105.83	3414.36
1000	12,626.53	229.37	12445.88	7319.52	859.59	6160.28	11,653.53	126.93	11,510.04	5041.21	283.99	4602.21
S03												
100	3103.65	305.91	2818.20	2296.08	27.50	2272.19	2557.78	28.80	2534.19	2271.25	19.12	2226.04
500	7989.73	231.91	7943.98	3251.82	188.68	3025.28	5938.44	431.01	5526.09	2776.84	257.98	2596.41
1000	10,899.55	178.64	10782.20	5801.48	980.81	4933.11	10,207.17	197.95	10,106.16	3821.32	213.93	3490.69
S04												
100	2827.21	28.44	2570.57	2058.26	22.21	2040.04	2300.50	30.77	2277.28	2047.22	19.14	2001.98
500	7282.02	125.93	7245.94	2932.21	181.69	2695.69	5396.79	401.24	5001.00	2450.00	68.07	2342.31
1000	11,657.35	120.64	11430.29	5164.56	846.35	4372.37	9070.54	289.83	8981.97	3481.64	199.58	3173.13
S05												
100	6371.82	64.81	5793.41	4638.79	50.06	4597.73	5184.74	69.34	5132.40	4769.02	45.04	4662.52
500	9847.10	35.06	9798.30	6608.46	409.49	6075.41	12163.00	904.29	11270.98	5695.82	160.20	5442.36
1000	13,957.38	144.45	13,685.52	11,639.61	1907.46	9854.21	20442.70	202.46	20,243.09	8095.69	469.75	7369.57
S06												
100	2830.04	87.72	2573.14	2561.93	27.46	2539.36	2300.50	30.77	2277.28	2516.30	23.55	2460.61
500	7296.57	125.98	7260.42	3651.08	227.14	3354.09	5396.79	401.24	5001.00	3005.36	84.88	2871.00
1000	9938.45	202.85	9744.86	6425.58	1051.11	5438.46	9070.54	389.83	8981.97	4267.90	246.92	3886.08

Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/ number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
100	2664.31	56.40	2114.40	2894.78	32.44	2253.74	2328.83	15.98	2289.20	2015.81	14.75	1978.97
500	7937.31	73.50	7818.66	8752.85	86.31	8582.09	2809.25	67.64	2683.47	2457.45	55.73	2310.31
1000	10,899.69	227.34	10,856.94	13,355.20	270.66	13,258.45	3787.14	144.97	3559.19	3072.06	117.87	3072.06
S02												
100	3087.49	38.80	2459.89	2860.83	37.08	2185.46	2884.51	18.63	2838.92	2921.66	20.67	2869.97
500	9174.56	163.24	9095.98	8752.85	186.31	8582.09	3490.96	89.46	3332.63	3684.95	83.56	3684.95
1000	13,547.70	217.78	13231.36	13,355.20	170.66	13,258.45	4727.66	191.94	4437.56	4606.56	176.75	4606.56
S03												
100	2570.01	40.45	2055.88	2784.76	73.27	2185.46	2257.75	15.06	2220.60	2195.41	14.98	2158.33
500	7937.31	63.50	7818.66	8752.85	86.31	8582.09	2727.65	67.54	2604.79	2674.94	64.78	2674.94
1000	10,899.69	227.34	10,856.94	13,355.20	270.66	13,258.45	3693.10	145.02	3468.95	3354.88	138.97	3354.88
S04												
100	2341.79	39.94	1828.11	2784.76	37.27	2185.46	2025.82	14.75	1788.99	2015.81	14.75	1978.97
500	7227.71	71.92	7085.41	8752.85	86.31	8582.09	2436.03	55.73	2328.33	2457.45	55.73	2310.30

Table 5 continued

Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/ number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
1000	9936.60	252.99	9864.03	13355.20	270.66	13,258.45	3296.23	117.88	3102.10	3072.06	117.87	3072.06
S05												
100	5277.80	72.07	4120.10	6178.64	84.50	4829.19	4565.69	33.25	4482.67	4029.61	29.49	3955.97
500	16,289.42	162.10	15968.72	19545.98	194.34	19161.48	5490.20	125.61	5247.46	4912.45	111.40	4912.45
1000	22,394.57	119.43	22,231.03	29,843.62	159.10	29625.76	7428.87	265.67	6991.34	6141.06	235.63	6141.06
S06												
100	2431.91	28.67	1845.91	2860.83	79.08	2185.46	2481.16	18.04	2436.12	2015.81	14.75	1978.97
500	7249.37	72.14	7106.65	8752.85	86.31	8582.09	2983.80	68.34	2851.83	2457.45	55.73	2457.45
1000	9966.38	253.15	9893.60	13355.20	270.66	13,258.45	4037.46	144.61	3799.56	3072.06	117.87	3072.06

S01 uniform distribution, S02 normal distribution, S03 left skewed, S04 right skewed, S05 HPC2N, S06 NASA

makespan of all tasks mapping on VMs as defined in Eq. (4).

5.2 Throughput

Throughput uses to estimate the total number of tasks, whose are executed successfully. In cloud computing, throughput means some tasks completed in a certain time period. Minimum throughput is required for task scheduling in cloud computing [55, 56]. This study decreases the throughput of total numbers of tasks mapping on VMs as defined in Eq. (5).

5.3 Degree of imbalance

Degree of imbalance (DI) describes the amount of load distribution amongst the VMs regarding to their execution competencies. It also uses to examine the unbalanced load on the VMs. A stable n average DI is considered for the better performance for resource scheduling [35, 55]. Although, proposed HGDCS algorithm calculates the balanced DI by using the Eq. (6).

5.4 Performance improvement rate

Performance improvement rate is used to estimate the percentage of performance improvement for the *i*th algorithm with regards to the *k*th algorithm. It always represents in percentage (%) [59].

$$PIR(\%) = \left(\frac{\left(\sum_{k^{th}} PM - \sum_{i^{th}} PM \right)}{\sum_{i^{th}} PM} \right) \times 100 \qquad (12)$$

where *PM* represents the specific performance metric for the evaluation.

6 Simulation setup

The simulation setup is described in this section along with the computational results and significance analysis obtained after the implementation of hybrid gradient descent cuckoo search (HGDCS) algorithm and other metaheuristic algorithms for optimization of resource scheduling in IaaS cloud computing. All algorithms are implemented in CloudSim simulator [60, 61] using two cases that are with using datasets and workload traces.

In order to get more precise results, the experimental setup is classified into two cases. This classification will cover the high and low demand of cloud users. Case-I is considered for the small scale of cloud computing. Here small scale means less number of users, cloudlets and VMs. Case-II is considered for large scale of cloud computing. Here large scale means the large number of users, cloudlets and VMs. Large cloudlets will enable the improvement perception in scalability with large problem size and fluctuating cloud user's demands in heterogeneous cloud computing environment.

Datasets are created based on uniform, normal, left-skewed and right-skewed distribution presented as S01, S02, S03 and S04 respectively. Uniform distribution consists of the equal amount of small, medium and large size tasks. Normal distribution signifies more medium size tasks, while small and large size tasks are less in amount. Skewness is the amount of asymmetric of the probability distribution of tasks in the datasets. It can be right (positive) skewed or left (negative) skewed. Furthermore, left skewed includes the more small and less large size tasks of the dataset. Hence the right skewed includes less small and large size task in the data sets. These datasets show the

behavior of the meta-heuristic algorithms for resource scheduling in cloud computing.

Workload traces (S05 and S06) are generated by the "Parallel Workload Archives" that consists of HPC2N (High-Performance Computing Center North) [62] and NASA Ames iPCS/860 [63]. These workload archives are provided by "Ake Sandgren" and "Bill Nitzberg", in the standard workload format (swf) acknowledged by the CloudSim tool. HPC2N comprises the statistics of 527,371 tasks and NASA comprises the statistics of 14,794 tasks. These workloads are mostly applied for evaluating the performance of algorithms in cloud computing environment [35, 36, 64–67].

The specification of cloud users, cloudlets, host, VMs and data center are shown in Tables 3 and 10 for Case-I and Case-II. These cases are implemented for simulation based on [36, 59, 68, 69]. The larger cloudlets will improve the perception in scalability of the performance of the algorithms with the large problem sizes and little user demand. The parameters setting of selected nature-inspired meta-heuristic algorithms are shown in Table 2. The parameters values of ACO is based on [28, 68], ABC is based on [70, 71], GA is based on [68, 72], LCA is based on [73], PSO is based on [39, 74], SA is based on [36] and CS is based on [75, 76]. These algorithms are compared with HGDCS algorithm, on a set of parameters including makespan, throughput, degree of imbalance and performance improving rate for resource scheduling in IaaS cloud computing. In the statistical analysis, mean, standard deviation and best value are mentioned after the 50 runs of the simulation, for the comparison of performance metrics, are shown in Tables 4, 5 and 6 in Case-I and Tables 11 to 13 in Case-II. Hence for the comparison of HGDCS algorithm's PIR % over existing algorithms is presented in Tables 7, 8 and 9 in the Case-I and Tables 14, 15 and 16 in the Case-II.

7 Results and discussion

This section presents and discusses the results of the simulation formulated in two cases with statistical significance analysis, to evaluate the performance of the proposed HGDCS algorithm for resource scheduling in IaaS cloud computing.

7.1 Case-I

In Case-1, we have fixed the specification of VMs to check the performance of HGDCS algorithm for evaluating the performance for resource scheduling, while changing the number of cloudlets with using datasets (S01, S02, S03 and S04) and workload traces (S05 and S06) in the simulations. Table 3 shows the simulation parameters setting of CloudSim for Case-I in cloud computing environment.

The comparison of makespan for resource scheduling among ACO, ABC, GA, LCA, PSO, SA and CS algorithms with HGDCS algorithm using the uniform distribution, normal distribution, left skewed, right skewed, HPC2 N and NASA in Case-I, is shown in Fig. 6. The x-axis represents the number of cloudlets while the y-axis represents the makespan. Figure 6(a) to (f) identify that makespan time of resource scheduling is increased with increasing the number of cloudlets. The comparison of simulation results clearly precise that the HGDCS algorithm delivers the less makespan time than other meta-heuristic algorithms with using all four datasets and two workload traces for the optimization of resource scheduling in IaaS cloud computing.

In Fig. 7, the comparison of throughput for resource scheduling is shown among ACO, ABC, GA, LCA, PSO, SA and CS algorithms with HGDCS algorithm using the uniform distribution, normal distribution, left skewed, right skewed, HPC2 N and NASA in Case-I. The horizontal axis signifies the number of cloudlets and the vertical axis signifies the throughput parameter. Figure 7(a) to (f) show that throughput of resource scheduling is increased with increasing the number of cloudlets. The comparison of simulation results expresses that the HGDCS algorithm provides the minimum throughput than other meta-heuristic algorithms with using all four datasets and two workload traces for the optimization of resource scheduling in IaaS cloud computing. Hence after the analysis of graphs, it is clearly distinct that HGDCS algorithm offers better results than all other comparison algorithms, while HGDCS algorithm results are near to the CS and LCA algorithms for the throughput.

The comparison of the degree of imbalance of imbalance for resource scheduling among ACO, ABC, GA, LCA, PSO, SA and CS algorithms with HGDCS algorithm using the uniform distribution, normal distribution, left skewed, right skewed, HPC2 N and NASA in Case-I, is shown in Fig. 8. The x-axis represents the number of cloudlets while the y-axis represents the degree of imbalance. The comparison of simulation results clearly demonstrates that the HGDCS algorithm provides the better degree of imbalance than other meta-heuristic algorithms due to the less fluctuation and move smoothly with increasing of cloudlets by using all four datasets and two workload traces for the optimization of resource scheduling in IaaS cloud computing.

Tables 4, 5 and 6 show that the mean, standard deviation and best values for makespan, throughput and degree of imbalance after the 50 runs at 100, 500 and 1000 number of cloudlets, are very close and significance from the values of the standard deviation by using all datasets and

Table 6 Statistical significance of meta-heuristic algorithms for degree of imbalance in Case-I after 50 runs

Algorithm	ACO			ABC			GA			LCA		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
100	0.5191	0.0737	0.5108	0.5291	0.0752	0.5208	0.5189	0.0741	0.5127	0.5256	0.0749	0.5190
500	0.5078	0.0755	0.5112	0.5210	0.0741	0.5131	0.5161	0.0735	0.5094	0.5228	0.0742	0.5143
1000	0.5068	0.0790	0.5218	0.5159	0.0738	0.5099	0.5956	0.0844	0.5851	0.5536	0.0786	0.5446
S02												
100	0.5857	0.0845	0.5957	0.5591	0.0807	0.5677	0.5845	0.0844	0.5923	0.5432	0.0784	0.5522
500	0.5246	0.0790	0.5084	0.5429	0.0784	0.5514	0.6239	0.0901	0.6341	0.5550	0.0801	0.5644
1000	0.5368	0.0836	0.5083	0.5694	0.0823	0.5713	0.5643	0.0814	0.5740	0.5618	0.0811	0.5712
S03												
100	0.5273	0.0761	0.5360	0.5372	0.0776	0.5459	0.5292	0.0764	0.5358	0.5264	0.0760	0.5332
500	0.5107	0.0752	0.5062	0.5296	0.0765	0.5378	0.5069	0.0732	0.5136	0.5143	0.0742	0.5228
1000	0.5213	0.0787	0.5056	0.5264	0.0761	0.5327	0.5827	0.0841	0.5932	0.5527	0.0798	0.5620
S04												
100	0.5023	0.0725	0.5089	0.4923	0.0711	0.4995	0.5003	0.0723	0.5046	0.4940	0.0713	0.5020
500	0.5526	0.0828	0.5401	0.4998	0.0721	0.5069	0.5226	0.0754	0.5305	0.5054	0.0730	0.5138
1000	0.5548	0.0877	0.5188	0.5029	0.0728	0.5031	0.5330	0.0769	0.5424	0.5120	0.0739	0.5204
S05												
100	0.5026	0.0726	0.5092	0.4926	0.0711	0.4998	0.5006	0.0723	0.5050	0.4944	0.0714	0.5024
500	0.5529	0.0828	0.5404	0.5001	0.0722	0.5072	0.5229	0.0755	0.5308	0.5064	0.0731	0.5148
1000	0.5552	0.0878	0.5192	0.5033	0.0728	0.5035	0.5334	0.0770	0.5428	0.5133	0.0741	0.5217
S06												
100	0.5023	0.0725	0.5089	0.4955	0.0715	0.5028	0.4980	0.0720	0.5023	0.4986	0.0720	0.5066
500	0.5593	0.0837	0.5471	0.5030	0.0726	0.5102	0.5081	0.0734	0.5157	0.5105	0.0737	0.5189
1000	0.5628	0.0887	0.5272	0.5061	0.0732	0.5065	0.5186	0.0749	0.5277	0.5174	0.0747	0.5258
Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
100	0.5149	0.0808	0.5208	0.5148	0.0801	0.5198	0.5265	0.0749	0.5190	0.5293	0.0764	0.5370
500	0.5016	0.0739	0.5039	0.5411	0.0802	0.5464	0.5165	0.0734	0.5085	0.5186	0.0734	0.5268
1000	0.5058	0.0793	0.5233	0.5375	0.0859	0.5627	0.5072	0.0720	0.4988	0.5085	0.0734	0.5170
S02												
100	0.5936	0.0905	0.5396	0.5112	0.0758	0.5111	0.5355	0.0773	0.5434	0.4943	0.0714	0.4995
500	0.6485	0.0947	0.6457	0.5464	0.0802	0.5411	0.5441	0.0785	0.5529	0.5049	0.0729	0.5135
1000	0.5184	0.0769	0.4998	0.5627	0.0859	0.5375	0.5369	0.0775	0.5461	0.5149	0.0743	0.5241
S03												
100	0.5372	0.0829	0.5317	0.5354	0.0821	0.5308	0.5354	0.0773	0.5433	0.5250	0.0758	0.5326
500	0.5039	0.0739	0.5016	0.5464	0.0802	0.5411	0.5250	0.0758	0.5334	0.5143	0.0758	0.5224
1000	0.5233	0.0793	0.5058	0.5627	0.0859	0.5375	0.5153	0.0744	0.5240	0.5042	0.0728	0.5126
S04												
100	0.5129	0.0762	0.5124	0.5112	0.0758	0.5111	0.4941	0.0713	0.4995	0.4944	0.0714	0.4997
500	0.5508	0.0811	0.5444	0.5464	0.0802	0.5411	0.5044	0.0728	0.5131	0.5051	0.0729	0.5138
1000	0.5678	0.0872	0.5402	0.5627	0.0859	0.5375	0.5140	0.0742	0.5232	0.5151	0.0743	0.5244
S05												
100	0.5132	0.0763	0.5128	0.5126	0.0760	0.5124	0.4944	0.0714	0.4998	0.4942	0.0714	0.4995
500	0.5511	0.0811	0.5447	0.5507	0.0810	0.5444	0.5048	0.0729	0.5135	0.5048	0.0728	0.5134

Table 6 continued

Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
1000	0.5682	0.0873	0.5405	0.5679	0.0872	0.5404	0.5144	0.0742	0.5237	0.5147	0.0743	0.5240
S06												
100	0.5076	0.0771	0.5044	0.5112	0.0758	0.5111	0.4935	0.0713	0.4996	0.4942	0.0714	0.4995
500	0.5891	0.0864	0.5839	0.5464	0.0802	0.5411	0.5037	0.0727	0.5124	0.5048	0.0727	0.5134
1000	0.6056	0.0920	0.5798	0.5627	0.0859	0.5375	0.5109	0.0737	0.5201	0.5147	0.0743	0.5240

workload. This significance analysis shows that results follow a normal distribution and robustness of the proposed HGDCS optimization algorithm and its capability to attain near optimum values in almost all runs.

Performance Improvement Rate (%) is based on makespan, throughput and degree of imbalance in Case-I of the CS algorithm as it relates to the ACO, ABC, GA, LCA, PSO, SA and CS meta-heuristic algorithms are presented in Tables 7, 8 and 9 respectively. For the makespan, HGDCS algorithm shows the 11.79% makespan improvement on CS algorithm and 47.74% makespan improvement on GA algorithm. In the case of throughput, HGDCS algorithm produces the 9.88% throughput improvement on CS algorithm while the 62.90% throughput improvement on ACO algorithm. In the same way, HGDCS algorithm gives the 1.10% for the improvement of degree of imbalance on CS algorithm and 7.59% for the improvements of degree of imbalance on PSO algorithm. This specifies that the HGDCS algorithm performs better in terms of makespan, throughput and degree of imbalance for resource scheduling in IaaS cloud computing.

7.2 Case-II

In Case-II, we have selected the specification of VMs randomly with different RAM, Bandwidth and MIPS to check the performance of HGDCS algorithm for evaluating the performance for resource scheduling, while changing the number of cloudlets with using datasets (S01, S02, S03 and S04) and workload traces (S05 and S06) in the simulations. Table 10 shows the simulation parameters setting of CloudSim for Case-II in cloud computing environment.

The comparison of makespan for resource scheduling among ACO, ABC, GA, LCA, PSO, SA and CS algorithms with HGDCS algorithm using the uniform distribution, normal distribution, left skewed, right skewed, HPC2 N and NASA in Case-II, is shown in Fig. 9. The x-axis represents the number of cloudlets while the y-axis represents the makespan. Figure 9(a) to (f) identify that makespan time of resource scheduling is increased with increasing the number of cloudlets. The comparison of simulation results clearly precise that the HGDCS algorithm delivers the less makespan time than other meta-heuristic algorithms with using all four datasets and two workload traces for the optimization of resource scheduling in IaaS cloud computing.

Table 7 Performance Improvement Rate (%) on makespan in Case-I

	ACO	ABC	GA	LCA	PSO	SA	CS	HGDCS
Total makespan	77,484.93	67,418.02	108,636.42	66,326.56	70,931.52	84,198.34	64,357.25	56,769.43
PIR over ACO	–	12.99	−40.20	14.40	8.46	−8.66	16.94	26.73
PIR over ABC	–	–	−61.14	1.62	−5.21	−24.89	4.54	15.79
PIR over GA	–	–	–	38.95	34.71	22.50	40.76	47.74
PIR over LCA	–	–	–	–	−6.94	−26.95	2.97	14.41
PIR over PSO	–	–	–	–	–	−18.70	9.27	19.97
PIR over SA	–	–	–	–	–	–	23.56	32.58
PIR over CS	–	–	–	–	–	–	–	11.79

Table 8 Performance improvement rate (%) on throughput in Case-I

	ACO	ABC	GA	LCA	PSO	SA	CS	HGDCS
Total throughput	874,391.65	462,264.03	782,414.69	371,228.28	909,731.00	1,092,522.95	356,972.17	189,192.7232
PIR over ACO	–	46.57	11.85	57.23	−2.47	−24.32	58.83	62.90
PIR over ABC	–	–	−64.97	19.95	−91.78	−132.68	22.95	30.56
PIR over GA	–	–	–	51.48	−16.25	−41.04	53.30	57.91
PIR over LCA	–	–	–	–	−139.59	−190.67	3.75	13.25
PIR over PSO	–	–	–	–	–	−21.32	59.82	63.79
PIR over SA	–	–	–	–	–	–	66.89	70.16
PIR over CS	–	–	–	–	–	–	–	9.88

Table 9 Performance improvement rate (%) on degree of imbalance in Case-I

	ACO	ABC	GA	LCA	PSO	SA	CS	HGDCS
Total DI	32.13	30.98	31.92	31.45	33.06	32.89	30.89	30.55
PIR over ACO	–	3.58	0.66	2.11	−2.89	−2.37	3.86	4.92
PIR over ABC	–	–	−3.03	−1.53	−6.71	−6.17	0.29	1.39
PIR over GA	–	–	–	1.46	−3.57	−3.05	3.23	4.29
PIR over LCA	–	–	–	–	−5.10	−4.57	1.79	2.88
PIR over PSO	–	–	–	–	–	0.51	6.56	7.59
PIR over SA	–	–	–	–	–	–	6.09	7.13
PIR over CS	–	–	–	–	–	–	–	1.10

In Fig. 10, the comparison of throughput for resource scheduling is shown among ACO, ABC, GA, LCA, PSO, SA and CS algorithms with HGDCS algorithm using the uniform distribution, normal distribution, left skewed, right skewed, HPC2 N and NASA in Case-II. The horizontal axis signifies the number of cloudlets and the vertical axis signifies the throughput parameter. Figure 10(a) to (f) show that throughput of resource scheduling is increased with increasing the number of cloudlets. The comparison of simulation results expresses that the HGDCS algorithm provides the minimum throughput than other meta-heuristic algorithms with using all four datasets and two workload traces for the optimization of resource scheduling in IaaS cloud computing. Hence after the analysis of graphs, it is clearly distinct that HGDCS algorithm offers better results than all other comparison algorithms, while HGDCS algorithm results are near to the CS and LCA algorithms for the throughput.

The comparison of the degree of imbalance of imbalance for resource scheduling among ACO, ABC, GA, LCA, PSO, SA and CS algorithms with HGDCS algorithm using the uniform distribution, normal distribution, left skewed, right skewed, HPC2 N and NASA in Case-II, is shown in Fig. 11. The x-axis represents the number of

Table 10 Simulation parameters setting of CloudSim for Case-II in cloud computing for resource scheduling

Sr. No	Entities	Parameters	Values
1	User	Number of users	100
		Number of brokers	10
2	Cloudlet	Number of cloudlets	1000–10,000
		Length	800,000
		File Size	600
3	Host	RAM	2048 MB
		Storage	1,000,000
		Bandwidth	10,000
4	VM	Number of VMs	50
		Type of policy	Time shared
		RAM	128 to 15,360 MB
		Bandwidth	128 to 15,360 MB
		MIPS	256 to 30,720
		Size	10000
		VMM	Xen
		Operating system	Linux
		Number of CPUs	2 on each
5	Data center	Number of data centers	5

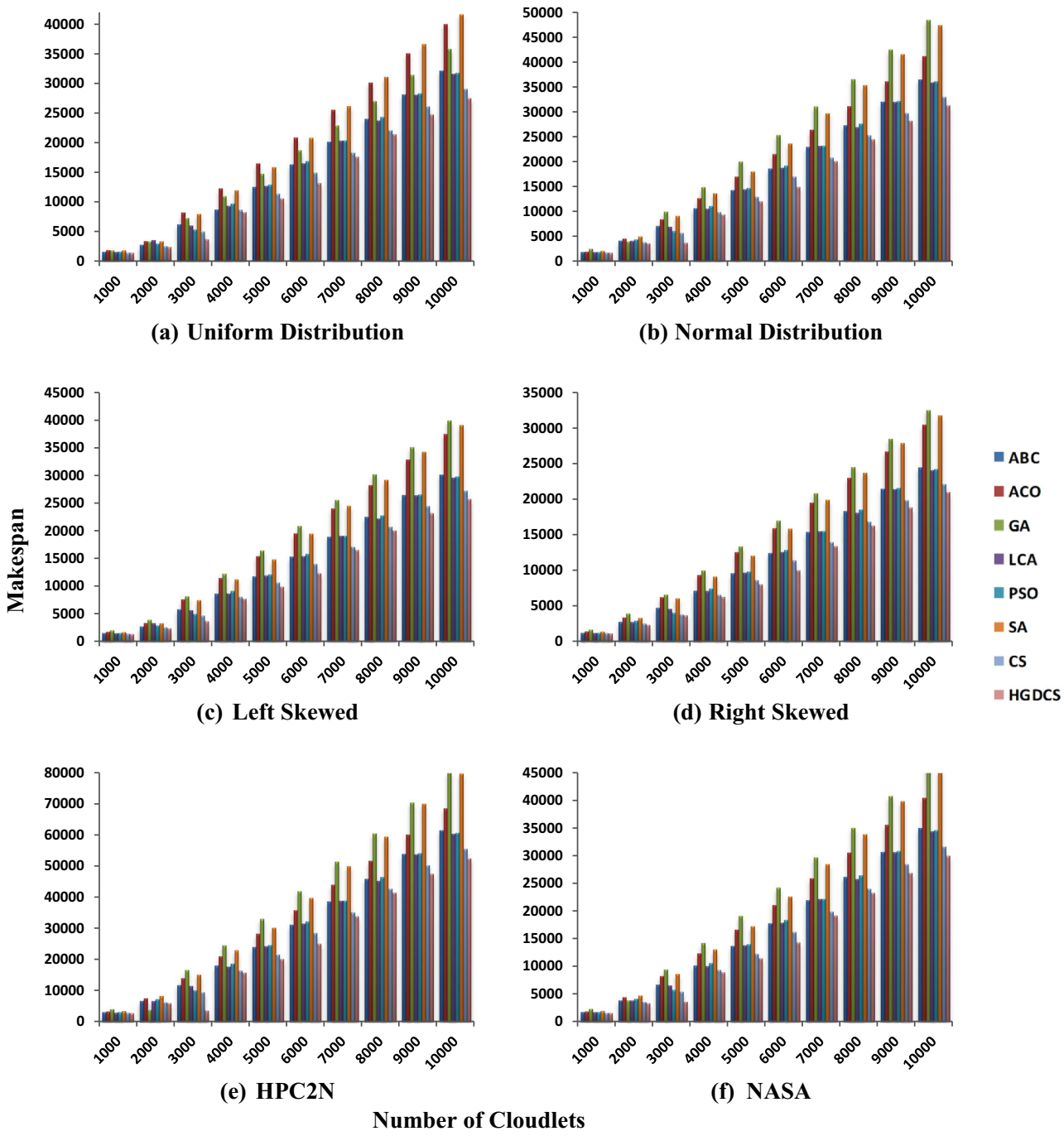

Fig. 9 Makespan for resource scheduling in Case-II using **a** uniform distribution, **b** normal distribution, **c** left skewed, **d** right skewed **e** HPC2N and **f** NASA

cloudlets while the y-axis represents the degree of imbalance. The comparison of simulation results clearly demonstrates that the HGDCS algorithm provides the better degree of imbalance than other meta-heuristic algorithms due to the less fluctuation and move smoothly with increasing of cloudlets by using all four datasets and

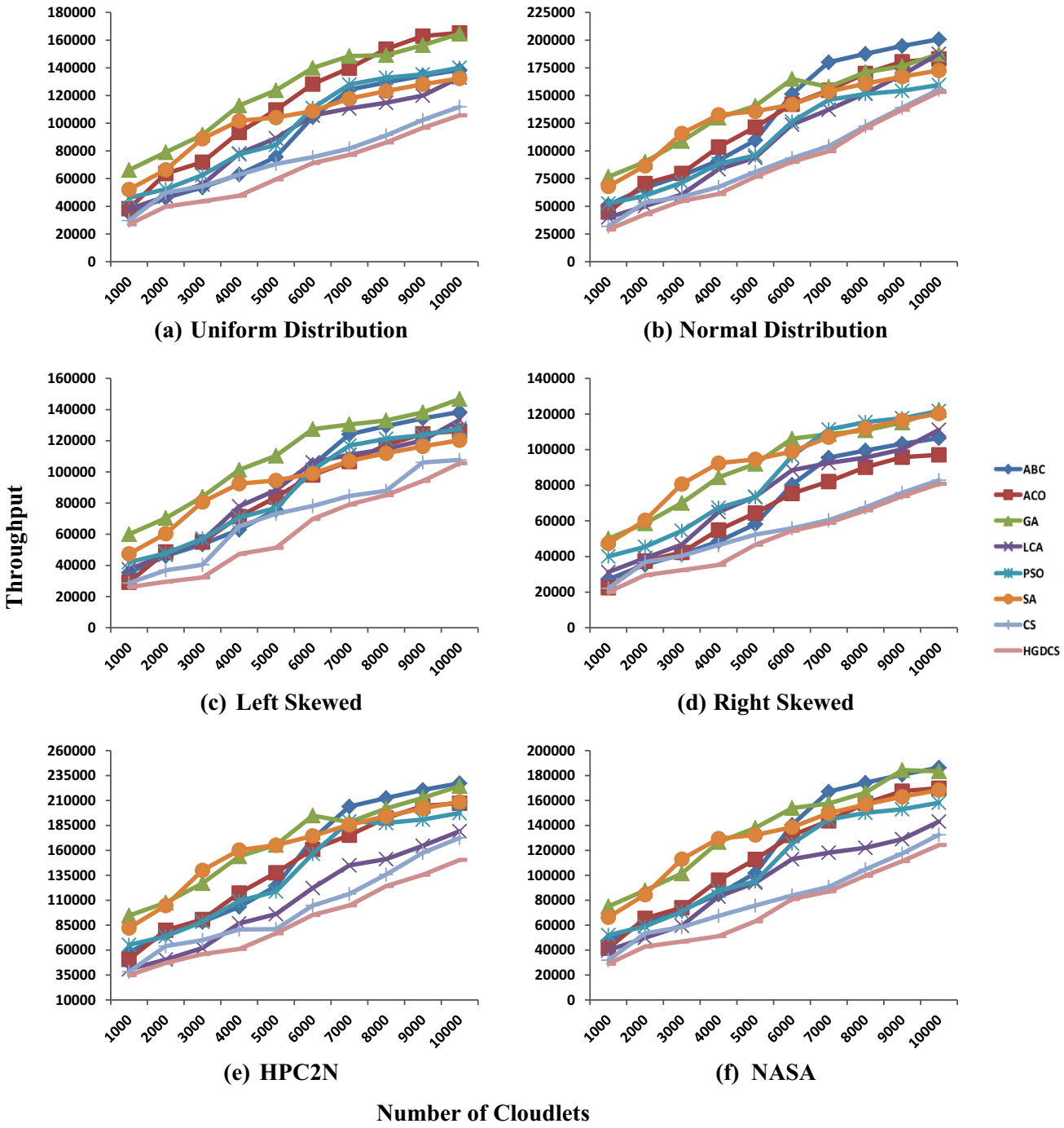

Fig. 10 Throughput for resource scheduling in Case-II using **a** uniform distribution, **b** normal distribution, **c** left skewed, **d** right skewed **e** HPC2N and **f** NASA

two workload traces for the optimization of resource scheduling in IaaS cloud computing.

The mean, standard deviation and best values for the 50 runs at 100, 500 and 1000 number of cloudlets of the makespan, throughput and degree of imbalance are computed and presented in Tables 11, 12 and 13. The results show that mean, standard deviation and best values are very close and significance from the values of the standard deviation by using all datasets and workload. This significance analysis shows that results follow a normal distribution and robustness of the proposed HGDCS optimization algorithm and its capability to attain near optimum values in almost all runs.

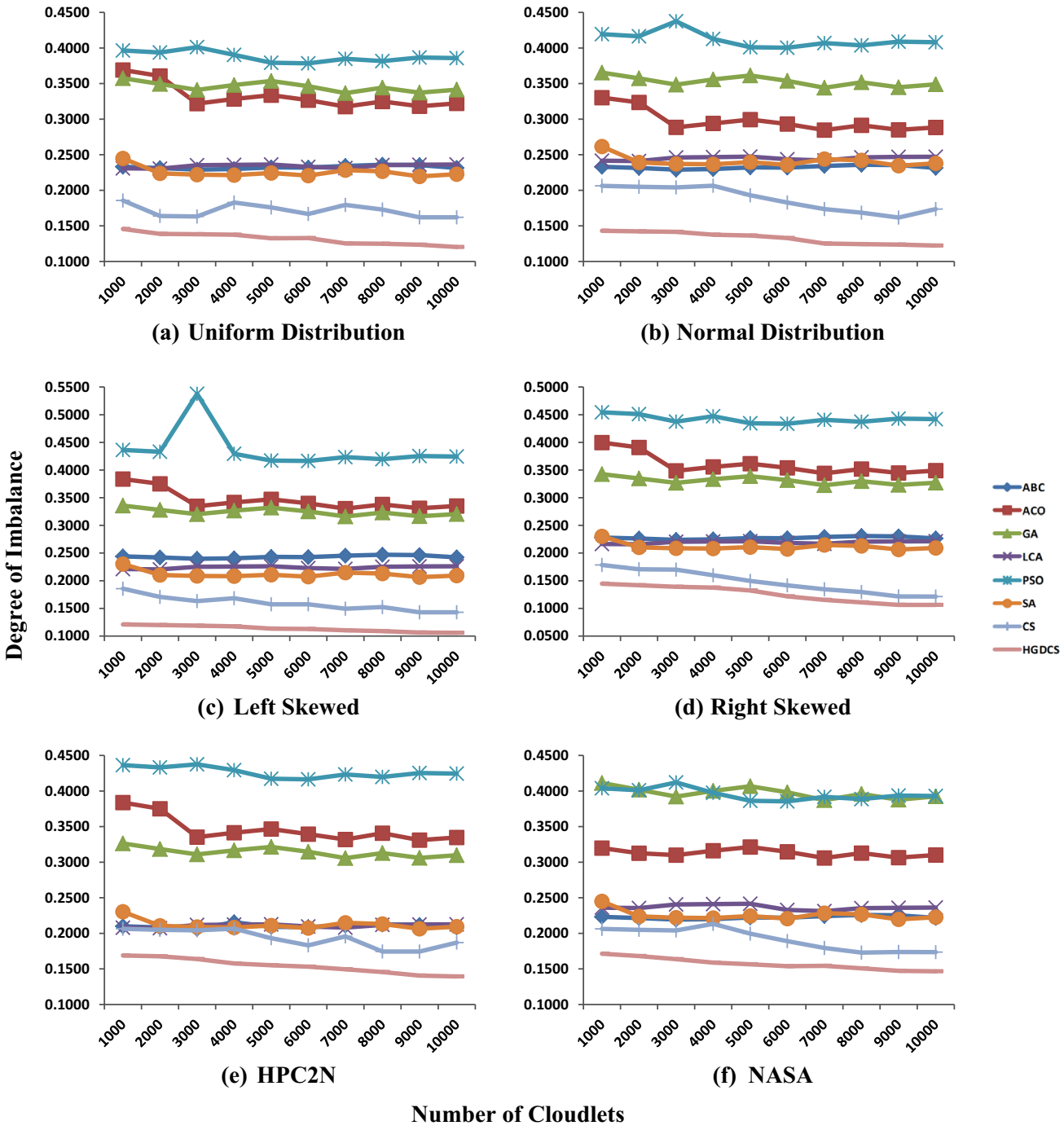

Fig. 11 Degree of imbalance for resource scheduling in Case-II using **a** uniform distribution, **b** normal distribution, **c** left skewed, **d** right skewed **e** HPC2N and **f** NASA

Table 11 Statistical significance of meta-heuristic algorithms for makespan in Case-II after 50 runs

Algorithm	ACO			ABC			GA			LCA		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
1000	1868.31	20.22	1837.59	1613.19	33.29	1550.12	1829.80	23.34	2143.86	1587.90	26.46	1551.83
5000	16,486.61	186.39	16170.15	12,581.28	212.20	12,271.82	14,766.97	202.48	17248.16	12,710.22	205.37	12,439.37
10000	39,992.35	370.14	39,588.04	32,156.90	504.43	31,314.63	35,845.07	399.27	42,227.24	31,609.96	1222.70	29,293.04
S02												
1000	1925.65	30.77	1893.70	1834.85	37.86	1763.12	2479.58	26.74	2438.43	1806.09	30.10	1765.05
5000	16,991.10	185.91	16,663.89	14,310.01	241.36	13,958.02	20,003.35	218.86	19,618.13	14,456.66	233.59	14,148.59
10000	41,198.91	389.42	40,796.83	36,575.41	573.74	35,617.41	48,502.82	458.45	48,029.45	35,953.31	590.71	33,318.04
S03												
1000	1754.14	18.94	1725.37	1514.68	31.25	1455.46	2046.50	22.10	2012.94	1490.93	24.85	1457.06
5000	15,476.11	174.35	15,182.66	11,812.96	199.24	11,522.39	16,507.85	185.97	16,194.84	11934.02	192.83	11,679.71
10000	37,543.36	351.26	37,170.45	30,193.12	473.62	29,402.29	40,046.25	374.67	39,648.48	29,679.58	1148.03	27,504.15
S04												
1000	1426.13	15.40	1402.74	1231.45	25.41	1183.30	1663.82	17.97	1636.53	1212.14	20.20	1184.60
5000	12,582.20	141.74	12,343.63	9604.03	161.98	9367.80	13421.02	151.19	13,166.53	9702.46	156.77	9495.70
10000	30,523.06	285.57	30,219.88	24,547.25	385.06	23,904.30	32,557.93	304.61	32,234.53	24,129.74	933.36	22,361.10
S05												
1000	3365.67	36.34	3310.47	3090.93	63.78	2970.08	4176.18	45.10	4107.70	3042.47	50.70	2973.35
5000	28,348.85	527.45	7513.95	24106.12	406.58	23513.18	33178.08	577.69	8524.29	24356.21	392.50	23,834.21
10000	68,716.27	1352.62	15,681.61	61,613.61	966.50	59,999.79	80,442.62	907.48	17790.21	60,565.65	1042.74	56,126.36
S06												
1000	1896.75	30.48	1865.65	1760.97	36.34	1692.12	2379.26	25.69	2340.24	1733.36	28.89	1693.98
5000	16,734.33	188.52	16,417.02	13,733.77	231.64	13,395.95	19,192.05	216.21	18,828.14	13,874.51	224.18	13,578.85
10000	40,595.67	379.81	40,192.43	35,102.57	550.64	34,183.15	46,557.84	435.60	46,095.38	34,505.53	334.71	31,976.37
Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
1000	1617.95	24.91	1575.93	1818.15	137.01	1557.85	1440.58	18.50	1410.19	1374.03	17.80	1343.64
5000	12,921.46	166.12	12,720.76	15,849.49	378.31	15,464.81	11,346.89	86.13	11,165.96	10,574.62	81.06	10,404.33
10000	31,783.15	556.56	30,678.89	41,686.62	921.65	40,054.43	29,044.06	149.54	28,781.79	27,530.01	142.41	27,280.22
S02												
1000	1840.26	28.33	1792.47	2067.97	55.83	1771.91	1773.94	22.78	1736.52	1691.98	21.85	1654.78
5000	14,696.93	188.95	14,468.65	18,027.28	430.29	17,589.75	12,906.00	114.12	12,700.21	12,027.62	92.20	11,813.93

Table 11 continued

Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
10000	36,150.30	633.04	34,894.31	47,414.55	948.29	45,558.09	33,034.96	169.89	32,737.16	31,312.86	161.74	31,029.25
S03												
1000	1519.14	23.39	1479.69	1707.11	28.64	1462.72	1416.77	18.20	1386.89	1351.32	18.06	1321.43
5000	12,132.36	155.98	11,943.92	14,881.58	355.20	14,520.40	10,653.95	80.87	10,484.06	9928.84	76.11	9,768.95
10000	29,842.19	522.57	28,805.37	39,140.87	865.37	37,608.36	27,270.38	240.41	27,024.12	25,848.78	133.72	25,614.25
S04												
1000	1235.08	19.02	1203.00	1387.90	104.59	1189.20	1190.56	15.29	1165.45	1135.56	14.99	1,110.45
5000	9,863.71	126.81	9710.50	12,098.85	288.78	11,805.20	8661.74	65.75	8523.63	8072.23	61.88	7,942.24
10000	24,261.95	424.86	23,419.00	31,821.84	703.55	30,575.90	22,171.04	114.15	21,970.83	21,015.27	108.71	20,824.64
S05												
1000	3100.04	47.73	3019.53	3483.62	62.51	2984.89	2988.31	38.38	2925.28	2850.26	37.83	2787.22
5000	24,757.91	318.29	24,373.36	30,368.10	724.85	29,631.05	21,740.98	165.03	21,394.31	20,261.29	155.32	19,935.02
10000	60,897.49	1066.39	58781.69	79,872.83	1165.92	76,745.51	55,659.08	969.29	55,201.07	52,527.64	256.46	52,321.45
S06												
1000	1766.16	27.19	1720.29	1984.69	49.56	1700.56	1702.51	21.86	1666.59	1623.85	20.77	1587.94
5000	14,105.10	181.34	13886.02	17301.35	412.96	16,881.44	12,386.30	94.02	12,188.79	11,543.28	88.49	11,357.40
10000	34,694.58	607.55	33,489.17	45,505.24	806.08	43,723.54	31,704.59	163.24	31,418.29	30,051.84	155.46	29,749.17

S01 uniform distribution, S02 normal distribution, S03 left skewed, S04 right skewed, S05 HPC2N, S06 NASA

Table 12 Statistical significance of meta-heuristic algorithms for throughput in Case-II after 50 runs

Algorithm	ACO			ABC			GA			LCA		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
1000	38,333.38	27,113.93	11,571.95	35,480.40	18,808.68	8148.35	66,231.50	34,229.19	24,686.83	37,912.07	9034.85	18,705.02
5000	109,596.50	34,625.51	68,261.26	75,652.08	27,494.33	35,728.06	123,706.02	28,595.43	113,768.77	89,225.59	21,285.85	82,513.90
10000	165,214.03	23,610.95	114,693.03	138,486.80	25,617.11	108,342.27	164,589.62	20,490.57	168,216.45	133,016.40	13,955.61	118,086.75
S02												
1000	44,803.89	11,349.54	11,815.10	51,227.91	27,055.57	12,372.35	76,609.86	32,464.77	23,683.93	40,020.54	16,959.41	34,349.09
5000	121,477.94	38,378.09	51,805.69	109,642.73	39,833.18	72,982.66	140,517.41	27,365.84	109,146.91	93,959.00	18,298.57	58,228.65
10000	183,080.10	25,102.37	157,096.30	200,745.54	37,124.52	161,430.44	187,533.09	49,490.36	161,382.65	187,588.63	39,496.13	119,472.13
S03												
1000	29,235.24	20,720.13	11,571.95	35,329.60	18,659.01	11,571.95	59,983.46	25,383.67	18,515.12	37,489.66	15,864.79	17,303.84
5000	83,777.89	26,467.65	68,261.26	75,615.68	27,471.16	68,261.26	110,405.64	21,299.64	85,326.58	88,324.51	17,039.71	40,157.69
10000	126,262.14	17,311.98	114,693.03	138,445.20	25,603.12	114,693.03	146,778.04	15,455.58	126,162.34	133,434.59	14,050.53	82,394.57
S04												
1000	22,488.64	15,938.56	9643.29	27,176.61	143,53.09	6267.96	49,986.22	21,153.06	15,429.27	31,241.39	13,220.66	1002.95
5000	64,444.53	20,359.73	56,884.38	58,165.91	21,131.66	27,483.12	92,004.70	17,749.70	71,105.48	73,603.76	14,199.76	30,890.53
10000	97,124.72	13,316.91	95,577.53	106,496.31	19,694.71	83,340.21	122,315.04	22,879.65	105,135.28	111,195.49	11,708.77	63,380.44
S05												
1000	50,746.06	12,854.78	22,372.35	58,022.07	30,643.84	23,382.09	91,582.47	38,809.68	28,312.71	40,020.54	16,959.41	38,904.67
5000	137,589.07	43,468.02	78,671.10	124,184.21	45,116.09	58,676.47	167,980.09	32,714.22	130,478.56	101,282.38	19,724.80	65,951.28
10000	207,361.28	28,431.60	155,122.33	227,369.62	42,048.20	177,931.35	224,184.51	23,299.55	192,923.24	180,258.34	18,734.29	135,317.24
S06												
1000	41,595.13	10,536.71	123,14.49	47,559.07	25,117.90	10,968.93	74,979.33	31,729.59	23,143.91	398,95.25	168,82.78	31,889.08
5000	112,777.93	35,629.52	726,41.36	10,1790.33	36,980.40	48,095.47	138,007.05	26,624.55	96,658.22	93,992.00	28,133.10	54,058.43
10000	169,968.26	23,304.59	123,008.28	186,368.54	34,465.74	145,845.37	183,472.55	19,319.48	157,702.92	143,108.59	35,069.19	110,915.77

Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
1000	46,135.41	11,591.57	26,199.69	52,149.24	12,775.42	32,340.32	29,760.32	8327.18	180,057.87	27,048.09	7570.11	16,409.49
5000	84,262.98	21,165.46	58,579.10	104,090.12	45,654.20	35,761.63	70,563.07	19,749.22	69,617.96	59,446.38	18,777.08	23,596.05
10000	139,975.18	28,376.83	128,060.24	132,389.61	30,935.64	118,008.48	111,887.71	22,979.09	105,430.57	105,865.34	21,702.47	52,344.05

Table 12 continued

Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S02												
1000	52,505.22	13,239.90	29,867.65	68,031.05	16,666.11	42,189.42	31968.61	9941.22	19,395.49	29,055.12	8182.06	17,625.01
5000	96,012.60	28,204.39	66,517.98	135,695.18	59,643.32	46,365.67	81016.85	22,675.03	22,524.32	76,768.32	21,478	21,091.75
10000	159,571.70	39,295.28	145,988.67	172,656.61	34,335.63	153,660.43	155279.59	31,822.64	76,986.90	153,199.84	31,398.33	75,951.15
S03												
1000	42,052.23	10,604.04	23,921.46	47,408.40	11,614.02	29,400.29	28,658.09	10,018.77	17,389.06	26,046.30	7289.78	15,801.73
5000	76,897.96	26,571.02	53,275.27	94,555.38	41,564.80	32,310.57	73,176.51	20,480.67	20,344.55	51,274.61	19,904.49	20,469.97
10000	127,803.42	27,648.41	116,924.57	12,0318.19	29,989.99	107,080.44	117,660.52	22,063.70	53,377.58	105,583.74	21,639.39	52,344.71
S04												
1000	40,049.74	10,099.08	22,782.34	47,408.40	11,614.02	29,400.29	22,044.68	6168.28	13,376.20	20,035.62	5607.52	12,155.18
5000	73,236.15	16,258.12	50,738.35	94,555.38	41,564.80	32,310.57	52,268.94	14,629.05	14,531.82	46,625.22	14,012.10	14,478.55
10000	121,717.54	27,284.20	111,356.73	120,318.19	29,989.99	107,080.44	82,815.78	16,972.07	41,059.68	80,740.51	16,547.77	40,028.31
S05												
1000	65,000.74	16,390.81	36,975.74	82,253.57	20,150.32	51,009.50	38,269.36	10,703.65	23,261.96	34,781.65	9730.31	21,138.55
5000	118,862.28	10,156.92	82,348.34	164,070.45	72,112.55	56,058.84	81,016.85	22,675.03	22,524.32	75,659.32	21,790.43	21,091.55
10000	197,547.57	11,822.25	18,0731.97	208,752.06	17,332.63	185,784.56	171,882.59	35,246.19	85,198.84	150,391.86	31,336.78	85,007.09
S06												
1000	52,064.67	13,128.81	29617.04	66371.76	16,259.62	41,160.41	31,964.79	8944.01	19,395.49	29,051.65	8130.77	17,625.01
5000	95,207.00	38,135.55	65,959.86	132,377.53	58,190.72	45,234.80	75,789.96	21,212.13	21,071.14	63,105.85	25,554.05	25,343.89
10000	158,232.81	29,469.45	144,763.75	168,445.47	13,985.98	149,912.62	132,505.25	27,155.32	65,695.49	124,298.98	25,475.05	61,623.07

S01 uniform distribution, S02 normal distribution, S03 left skewed, S04 Right skewed, S05 HPC2N, S06 NASA

Table 13 Statistical significance of meta-heuristic algorithms for degree of imbalance in Case-II after 50 runs

Algorithm	ACO			ABC			GA			LCA		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
1000	0.3838	0.0576	0.3136	0.2098	0.0307	0.2054	0.2755	0.0536	0.2044	0.2078	0.0305	0.1844
5000	0.3471	0.0526	0.3178	0.2090	0.0308	0.2018	0.2727	0.0526	0.2102	0.2126	0.0310	0.1205
10000	0.3350	0.0509	0.3094	0.2083	0.0305	0.2042	0.2631	0.0509	0.2037	0.2126	0.0316	0.0975
S02												
1000	0.3830	0.0575	0.3136	0.2098	0.0307	0.2054	0.3283	0.0493	0.2044	0.2078	0.0305	0.1844
5000	0.3472	0.0485	0.3178	0.2090	0.0308	0.2019	0.3255	0.0485	0.2103	0.2126	0.0310	0.1205
10000	0.3345	0.0466	0.3094	0.2083	0.0305	0.2042	0.3136	0.0466	0.2037	0.2126	0.0316	0.0975
S03												
1000	0.3876	0.0576	0.3136	0.2119	0.0307	0.2054	0.3323	0.0493	0.2043	0.2099	0.0305	0.1843
5000	0.3506	0.0484	0.3178	0.2111	0.0308	0.2018	0.3286	0.0484	0.2102	0.2147	0.0310	0.1205
10000	0.3384	0.0468	0.3094	0.2104	0.0305	0.2042	0.3172	0.0468	0.2037	0.2147	0.0316	0.0975
S04												
1000	0.3997	0.0159	0.3135	0.2184	0.0051	0.2053	0.3426	0.0136	0.2043	0.2164	0.0054	0.1843
5000	0.3615	0.0120	0.3177	0.2176	0.0063	0.2018	0.3389	0.0120	0.2102	0.2214	0.0042	0.1204
10000	0.3489	0.0120	0.3094	0.2170	0.0051	0.2041	0.3271	0.0120	0.2036	0.2214	0.0079	0.0974
S05												
1000	0.3839	0.0576	0.2056	0.2099	0.0307	0.1852	0.3264	0.0534	0.2045	0.2080	0.0305	0.2066
5000	0.3466	0.0517	0.2019	0.2091	0.0308	0.1225	0.3215	0.0563	0.2103	0.2127	0.0310	0.2075
10000	0.3348	0.0500	0.2043	0.2084	0.0305	0.0976	0.3098	0.0560	0.2037	0.2127	0.0316	0.2094
S06												
1000	0.5482	0.0161	0.3189	0.2997	0.0051	0.2066	0.4699	0.0138	0.2091	0.2969	0.0053	0.1638
5000	0.4958	0.0122	0.3129	0.2986	0.0061	0.2101	0.4648	0.0122	0.2053	0.3037	0.0043	0.1139
10000	0.4786	0.0114	0.3136	0.2976	0.0052	0.2054	0.4487	0.0114	0.2024	0.3037	0.0037	0.1844
Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
S01												
1000	0.4362	0.0646	0.4231	0.2302	0.0410	0.2258	0.1856	0.0272	0.1765	0.1457	0.0227	0.1374
5000	0.4172	0.0609	0.4136	0.2108	0.0307	0.2090	0.1197	0.0173	0.1076	0.1325	0.0144	0.1307
10000	0.4245	0.0630	0.4167	0.2093	0.0310	0.2000	0.0972	0.0141	0.0893	0.1203	0.0133	0.1175
S02												
1000	0.4363	0.0646	0.4231	0.2344	0.0410	0.2246	0.1857	0.0272	0.1777	0.1432	0.0255	0.1401
5000	0.4172	0.0609	0.4136	0.2109	0.0307	0.2180	0.1197	0.0177	0.1155	0.1365	0.0157	0.1281
10000	0.4245	0.0630	0.4167	0.2084	0.0310	0.2050	0.0972	0.0154	0.0955	0.1223	0.0144	0.1142
S03												
1000	0.4406	0.0646	0.4231	0.2325	0.0410	0.2228	0.1875	0.0256	0.1755	0.1211	0.0231	0.1173
5000	0.4214	0.0609	0.4136	0.2130	0.0307	0.2089	0.1209	0.0169	0.1176	0.1135	0.0155	0.1105
10000	0.4287	0.0630	0.4167	0.2115	0.0310	0.2085	0.0982	0.0141	0.0899	0.1059	0.0137	0.0975
S04												
1000	0.4543	0.0148	0.4230	0.2397	0.0246	0.2355	0.1933	0.0249	0.1822	0.1447	0.0233	0.1344
5000	0.4345	0.0092	0.4136	0.2196	0.0043	0.2147	0.1246	0.0212	0.1199	0.1314	0.0217	0.1280
10000	0.4421	0.0147	0.4167	0.2180	0.0069	0.2110	0.1012	0.0210	0.0993	0.1063	0.0207	0.1023
S05												
1000	0.4364	0.0646	0.4233	0.3660	0.0411	0.2303	0.1858	0.0272	0.1845	0.1714	0.0233	0.1685
5000	0.4173	0.0609	0.4137	0.3332	0.0307	0.2109	0.1198	0.0173	0.1106	0.1565	0.0117	0.1481

Table 13 continued

Algorithm	PSO			SA			CS			HGDCS		
Statistical dispersion/number of cloudlets	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best	$\bar{x}$	σ	Best
10000	0.4245	0.0630	0.4168	0.3278	0.0310	0.2094	0.0973	0.0141	0.0956	0.1466	0.0107	0.1424
S06												
1000	0.6232	0.0143	0.4273	0.3289	0.0250	0.3122	0.2653	0.0349	0.2068	0.1690	0.0293	0.1647
5000	0.5960	0.0091	0.4135	0.3012	0.0044	0.2938	0.1710	0.0312	0.1669	0.1552	0.0277	0.1523
10000	0.6064	0.0143	0.4131	0.2991	0.0067	0.2758	0.1389	0.0310	0.1265	0.1394	0.0207	0.1225

S01 uniform distribution, S02 normal distribution, S03 left skewed, S04 right skewed, S05 HPC2N, S06 NASA

Table 14 Performance improvement rate (%) on makespan in Case-II

	ACO	ABC	GA	LCA	PSO	SA	CS	HGDCS
Total makespan	1,259,749.58	1,054,483.83	1,377,928.31	1,051,228.28	1,061,096.86	1,356,676.35	960,867.96	898,285.85
PIR over ACO	–	16.29	−9.38	16.55	15.77	−7.69	23.73	28.69
PIR over ABC	–	–	−30.67	0.31	−0.63	−28.66	8.88	14.81
PIR over GA	–	–	–	23.71	22.99	1.54	30.27	34.81
PIR over LCA	–	–	–	–	−0.94	−29.06	8.60	14.55
PIR over PSO	–	–	–	–	–	−27.86	9.45	15.34
PIR over SA	–	–	–	–	–	–	29.17	33.79
PIR over CS	–	–	–	–	–	–	–	6.51

Table 15 Performance improvement rate (%) on throughput in Case-II

	ACO	ABC	GA	LCA	PSO	SA	CS	HGDCS
Total throughput	6,477,793.19	6,518,675.90	7,701,770.95	5,675,931.80	6,277,924.25	7,139,316.41	4,723,356.45	4,260,323.12
PIR over ACO	–	−0.63	−18.89	12.38	3.09	−10.21	27.08	34.23
PIR over ABC	–	–	−18.15	12.93	3.69	−9.52	27.54	34.64
PIR over GA	–	–	–	26.30	18.49	7.30	38.67	44.68
PIR over LCA	–	–	–	–	−10.61	−25.78	16.78	24.94
PIR over PSO	–	–	–	–	–	−13.72	24.76	32.14
PIR over SA	–	–	–	–	–	–	33.84	40.33
PIR over CS	–	–	–	–	–	–	–	9.80

In the Case-II, Performance Improvement Rate (%) is based on makespan, throughput and degree of imbalance of the HGDCS algorithm as it compares to the ACO, ABC, GA, LCA, PSO, SA and CS meta-heuristic algorithms are presented in Tables 14, 15 and 16 correspondingly. In the case of makespan, HGDCS algorithm shows the 6.51% makespan improvement on CS algorithm and the 33.79% makespan improvements on SA algorithm. Similarly, HGDCS algorithm produces the 9.80% throughput improvement on CS algorithm, also the 44.68% makespan improvement on GA algorithm. In the same way, HGDCS algorithm offers the 22.38% for the improvements of degree of imbalance on CS algorithm and the 67.41% for the improvements of degree of imbalance on GA algorithm. This identifies that the HGDCS algorithm performs better in terms of makespan, throughput and degree of imbalance for resource scheduling in IaaS cloud computing.

The novelty of the proposed approach in terms of local search trap, global search tarp, processing speed, selection criteria, scalability and robustness are shown in Table 17. The processing speed is based on execution of task successfully and measuring scale is as (Fastest: 0–10 min, Fast: 10–25 min, Medium: 25–1 h and Slow: 1 h and

Table 16 Performance improvement rate (%) on degree of imbalance in Case-II

	ACO	ABC	GA	LCA	PSO	SA	CS	HGDCS
Total makespan	19.95	13.68	20.66	13.70	25.02	13.28	10.51	8.15
PIR over ACO	–	31.41	−3.57	31.34	−25.44	33.45	47.33	59.12
PIR over ABC	–	–	−51.00	−0.10	−82.88	2.97	23.22	40.40
PIR over GA	–	–	–	33.71	−21.11	35.74	49.15	60.53
PIR over LCA	–	–	–	–	−82.69	3.07	23.29	40.46
PIR over PSO	–	–	–	–	–	46.94	58.01	67.41
PIR over SA	–	–	–	–	–	–	20.86	38.57
PIR over CS	–	–	–	–	–	–	–	22.38

Table 17 Significance analysis of meta-heuristic algorithms for resource scheduling in IaaS cloud computing

Algorithms Features	ACO	ABC	GA	LCA	PSO	SA	CS	HGDCS
Local search trap	Yes	Yes	No	No	No	No	No	No
Global search trap	No	No	No	No	No	Yes	No	No
Processing speed	Slow	Medium	Slow	Fast	Medium	Medium	Fast	Fastest
Selection criteria	Random selection	Random selection	Random selection	Random selection	Random selection	Random selection	Random walk	Gradient and random isotropic
Scalability	No	No	No	Yes	No	No	Yes	Yes
Robustness	Medium	Medium	Medium	Medium	Low	Medium	High	Highest

more). The robustness measure in terms of standard deviation with measuring scale is as (Highest: 0–10 values, High: 10–20 values, Medium: 20–30 values and Low: 30 or more values). The results in Table 17 show that the proposed HGDCS algorithm competes all the features deliberated in Table 17 with the previous meta-heuristic algorithms. However, HGDCS algorithm outperforms in terms of processing speed, and robustness.

The HGDCS algorithm uses the beneficial information of the best solution, to accelerate the convergence speed and enhance its capacity for handling the optimization of resource scheduling problematic issue in IaaS cloud computing. The results obtained using the HGDCS algorithm for resource scheduling may yield better solutions than other existing meta-heuristic algorithms, which demonstrate the effectiveness and robustness of HGDCS algorithm. After the performance evaluation, we suggest that the HGDCS algorithm is potentially a powerful search and optimization technique for resolving the complex problematic issues of resource scheduling in IaaS cloud computing.

The efficient searching of HGDCS algorithm due to the GD approach supporting the local search and improves the convergence rate. Due to the ideal features of HGDCS algorithm including the satisfaction of global search through Levy Flights, supporting the global search provide the optimal solution for resource scheduling in both scenarios. Moreover, there are still open loopholes are remain for improving HGDCS algorithm, including how to design self-adaptive, alteration of parameters and control these achievements for best performance. The simulation results and statistical analysis show that the proposed HGDCS algorithm is significantly performed better than the original CS algorithm and rest of the compared algorithms according to achieve the high performance. Therefore, it will be very useful to carry out large world scale applications for resource scheduling in IaaS cloud computing.

8 Conclusion and recommendations

Scheduling of resources is an NP-hard problem in IaaS cloud computing environment. The execution of tasks silently relies on the schedule of resources to be executed effectively. This paper provides the overview of resource scheduling problem, propose the HGDCS algorithm and

discusses it with respect to convergence speed and performance. This paper also presents a performance comparison of meta-heuristic algorithms, which includes the ACO, ABC, GA, LCA, PSO, SA and original CS algorithms with HGDCS algorithm for solving and optimizing the resource scheduling problems in IaaS cloud computing. This paper compares the makespan, throughput, degree of imbalance and performance improvement rate of each algorithm applicable for cloud computing system. Simulation results are presented with the help of graphical representation and statistical analysis, which indicate that proposed HGDCS algorithm is efficient and outperform than existing meta-heuristic algorithms for optimal resource scheduling in cloud computing environment. Time/Space complexity, additional cases and comparison of the proposed algorithm with existing meta-heuristic algorithms are required. In the future, we will enhance the CS algorithms for multi-objective resource scheduling in IaaS cloud computing. It can also help to provide solutions for the certain problem in cloud computing like decision making, security, green computing (heat and energy consumption), big data (storage), etc.

References

1. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Grid Computing Environments Workshop, 2008. GCE'08 2008, pp. 1–10. IEEE
2. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. ACM SIGCOMM Comput Commun Rev **39**(1), 50–55 (2008)
3. Gill, G.S., Wadhwa, A., Jatain, A.: Cloud computing: a new age of computing. In: 2014 fourth international conference on advanced computing & communication technologies 2014, pp. 243–250. IEEE
4. Shojafar, M., Canali, C., Lancellotti, R., Abawajy, J.: Adaptive computing-plus-communication optimization framework for multimedia processing in cloud systems. IEEE Trans. Cloud Comput. 1–14 (2016)
5. Canali, C., Lancellotti, R.: Automatic parameter tuning for class-based virtual machine placement in cloud infrastructures. In: Software, Telecommunications and Computer Networks (SoftCOM), 2015 23rd International Conference on 2015, pp. 290–294. IEEE
6. Younas, M., Ghani, I., Jawawi, D.N., Khan, M.M.: A Framework for agile development in cloud computing environment. 인터넷정보학회논문지 **17**(5), 67–74 (2016)
7. Madni, S.H.H., Latiff, M.S.A., Coulibaly, Y.: Resource scheduling for infrastructure as a service (IaaS) in cloud computing: challenges and opportunities. J. Netw. Comput. Appl. **68**, 173–200 (2016)
8. Tsai, C.-W., Rodrigues, J.J.: Metaheuristic scheduling for cloud: a survey. IEEE Syst. J. **8**(1), 279–291 (2014)
9. Mathew, T., Sekaran, K.C., Jose, J.: Study and analysis of various task scheduling algorithms in the cloud computing environment. In: Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on 2014, pp. 658–664. IEEE
10. Thaman, J., Singh, M.: Current perspective in task scheduling techniques in cloud computing: a review. Int. J. Found. Comput. Sci. Technol. **6**, 65–85 (2016)
11. Kalra, M., Singh, S.: A review of metaheuristic scheduling techniques in cloud computing. Egypt. Inform. J. **16**(3), 275–295 (2015)
12. Madni, S.H.H., Latiff, M.S.A., Coulibaly, Y., Abdulhamid, S.I. M.: An appraisal of meta-heuristic resource allocation techniques for IaaS cloud. Indian J. Sci. Technol. **9**(4), 1–14 (2016)
13. Hallaj, E., Tabbakh, S.R.K.: Study and analysis of task scheduling algorithms in clouds based on artificial bee colony. In: Technology, Communication and Knowledge (ICTCK), 2015 International Congress on 2015, pp. 38–45. IEEE
14. Huang, M.G., Ou, Z.Q.: Review of task scheduling algorithm research in cloud computing. Adv. Mater. Res. **926**, 3236–3239 (2014)
15. Singh, P., Dutta, M., Aggarwal, N.: A review of task scheduling based on meta-heuristics approach in cloud computing. Knowl. Inf. Syst. **52**(1), 1–51 (2017)
16. Cui, Y.F., Li, X.M., Dong, K.W., Zhu, J.L.: Cloud computing resource scheduling method research based on improved genetic algorithm. Adv. Mater. Res. **271**, 552–557 (2011)
17. Chen, S., Wu, J., Lu, Z.: A cloud computing resource scheduling policy based on genetic algorithm with multiple fitness. In: Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on 2012, pp. 177–184. IEEE
18. Sindhu, S., Mukherjee, S.: A genetic algorithm based scheduler for cloud environment. In: Computer and Communication Technology (ICCCT), 2013 4th International Conference on 2013, pp. 23–27. IEEE
19. Javanmardi, S., Shojafar, M., Amendola, D., Cordeschi, N., Liu, H., Abraham, A.: hybrid job scheduling algorithm for cloud computing environment. In: Proceedings of the Fifth international conference on innovations in bio-inspired computing and applications IBICA 2014 2014, pp. 43–52. Springer
20. Shojafar, M., Javanmardi, S., Abolfazli, S., Cordeschi, N.: FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Clust. Comput. **18** (2), 829–844 (2015)
21. Saha, S., Pal, S., Pattnaik, P.K.: A novel scheduling algorithm for cloud computing environment. In: Computational Intelligence in Data Mining—Vol. 1, pp. 387–398. Springer (2016)
22. Zhang, H., Li, P., Zhou, Z., Yu, X.: A PSO-based hierarchical resource scheduling strategy on cloud computing. In: Trustworthy Computing and Services. pp. 325–332. Springer (2013)
23. Netjinda, N., Sirinaovakul, B., Achalakul, T.: Cost optimal scheduling in IaaS for dependent workload with particle swarm optimization. J. Supercomput. **68**(3), 1579–1603 (2014)
24. Liu, J., Luo, X.G., Zhang, X.M., Zhang, F.: Job scheduling algorithm for cloud computing based on particle swarm optimization. Adv. Mater. Res. **662**, 957–960 (2013)
25. Abdi, S., Motamedi, S.A., Sharifian, S.: Task scheduling using Modified PSO Algorithm in cloud computing environment. In: International Conference on Machine Learning, Electrical and Mechanical Engineering, pp. 8–9 (2014)
26. Al-Olimat, H.S., Alam, M., Green, R., Lee, J.K.: Cloudlet scheduling with particle swarm optimization. In: Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on 2015, pp. 991–995. IEEE
27. Wang, G., Yu, H.C.: Task scheduling algorithm based on improved min–min algorithm in cloud computing environment. Appl. Mech. Mater. **303**, 2429–2432 (2013)
28. Li, K., Xu, G., Zhao, G., Dong, Y., Wang, D.: Cloud task scheduling based on load balancing ant colony optimization. In:

Chinagrid Conference (ChinaGrid), 2011 Sixth Annual 2011, pp. 3–9. IEEE
29. Tawfeek, M.A., El-Sisi, A., Keshk, A.E., Torkey, F.A.: Cloud task scheduling based on ant colony optimization. In: Computer Engineering & Systems (ICCES), 2013 8th International Conference on 2013, pp. 64–69. IEEE
30. Wen, X., Huang, M., Shi, J.: Study on resources scheduling based on ACO allgorithm and PSO algorithm in cloud computing. In: Distributed Computing and Applications to Business, Engineering & Science (DCABES), 2012 11th International Symposium on 2012, pp. 219–222. IEEE
31. Yang, H.: Improved ant colony algorithm based on PSO and its application on cloud computing resource scheduling. Adv. Mater. Res. **989**, 2192–2195 (2014)
32. Cho, K.-M., Tsai, P.-W., Tsai, C.-W., Yang, C.-S.: A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing. Neural Comput. Appl. **26**(6), 1297–1302 (2014)
33. Liu, C.-Y., Zou, C.-M., Wu, P.: A Task Scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing. In: Distributed computing and applications to business, engineering and science (DCABES), 2014 13th International Symposium on 2014, pp. 68–72. IEEE
34. Muthulakshmi, B., Somasundaram, K.: A hybrid ABC-SA based optimized scheduling and resource allocation for cloud environment. Clust. Comput. (2017). https://doi.org/10.1007/s10586-017-1174-z
35. Abdullahi, M., Ngadi, M.A., Abdulhamid, S.M.: Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Gener. Comput. Syst. **56**, 640–650 (2016)
36. Abdullahi, M., Ngadi, M.A.: Hybrid symbiotic organisms search optimization algorithm for scheduling of tasks on cloud computing environment. PLoS ONE **11**(6), e0158229 (2016)
37. Tsai, J.-T., Fang, J.-C., Chou, J.-H.: Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. Comput. Oper. Res. **40** (12), 3045–3055 (2013)
38. Guddeti, R.M., Buyya, R.: A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment. IEEE Transactions on Services Computing (2017)
39. Gabi, D., Ismail, A.S., Zainal, A., Zakaria, Z., Abraham, A.: Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing. Neural Comput. Appl. (2016). https://doi.org/10.1007/s00521-016-2816-4
40. Moon, Y., Yu, H., Gil, J.-M., Lim, J.: A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments. Human-Centric Comput. Inf. Sci. **7**(1), 28 (2017). https://doi.org/10.1186/s13673-017-0109-2
41. Gill, S.S., Buyya, R., Chana, I., Singh, M., Abraham, A.: BULLET: particle swarm optimization based scheduling technique for provisioned cloud resources. J. Netw. Syst. Manage. **26**(2), 361–400 (2018). https://doi.org/10.1007/s10922-017-9419-y
42. Snyman, J.: Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms, vol. 97. Springer Science & Business Media, Berlin (2005)
43. Fletcher, R., Powell, M.J.: A rapidly convergent descent method for minimization. Comput. J. **6**(2), 163–168 (1963)
44. Yang, X.-S., Deb, S.: Cuckoo search via Lévy flights. In: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on 2009, pp. 210–214. IEEE
45. Yang, X.-S.: Cuckoo search and firefly algorithm: overview and analysis. In: Cuckoo Search and Firefly Algorithm. pp. 1–26. Springer (2014)
46. Burnwal, S., Deb, S.: Scheduling optimization of flexible manufacturing system using cuckoo search-based approach. Int. J. Adv. Manuf. Technol. **64**(5–8), 951–959 (2013)
47. Gunavathi, C., Premalatha, K.: Cuckoo search optimisation for feature selection in cancer classification: a new approach. Int. J. Data Min. Bioinform. **13**(3), 248–265 (2015)
48. Majumder, A., Laha, D.: A new cuckoo search algorithm for 2-machine robotic cell scheduling problem with sequence-dependent setup times. Swarm Evolut. Comput. **28**, 131–143 (2016)
49. Wang, H., Wang, W., Sun, H., Cui, Z., Rahnamayan, S., Zeng, S.: A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems. Soft Comput. **21**(15), 4297–4307 (2016)
50. Zendaoui, Z., Layeb, A.: Adaptive Cuckoo Search Algorithm for the Bin Packing Problem, pp. 107–120. Springer, Berlin (2016)
51. Civicioglu, P., Besdok, E.: A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artif. Intell. Rev. **39**(4), 315–346 (2013)
52. Civicioglu, P., Besdok, E.: Comparative analysis of the cuckoo search algorithm. In: Yang, S. (ed.) Cuckoo Search and Firefly Algorithm, pp. 85–113. Springer, Cham (2014)
53. Gandomi, A.H., Yang, X.-S., Alavi, A.H.: Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng. Comput. **29**(1), 17–35 (2013)
54. Gandomi, A.H., Yang, X.-S., Talatahari, S., Deb, S.: Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization. Comput. Math. Appl. **63**(1), 191–200 (2012)
55. Madni, S.H.H., Latiff, M.S.A., Coulibaly, Y., Abdulhamid, S.I. M.: Recent advancements in resource allocation techniques for cloud computing environment: a systematic review. Clust. Comput. (2016). https://doi.org/10.1007/s10586-016-0684-4
56. Mustafa, S., Nazir, B., Hayat, A., Madani, S.A.: Resource management in cloud computing: taxonomy, prospects, and challenges. Comput. Electr. Eng. **47**, 186–203 (2015)
57. Abdulhamid, S.M., Latiff, M.S.A., Idris, I.: Tasks Scheduling technique using league championship algorithm for makespan minimization in IaaS cloud. ARPN J. Eng. Appl. Sci. **9**(12), 2528–2533 (2015)
58. Madni, S.H.H., Latiff, M.S.A., Abdulhamid, S.I.M.: Optimal resource scheduling for IaaS cloud computing using cuckoo search algorithm. Sains Humanika **9**(1–3), 71–76 (2017)
59. Abdulhamid, S.I.M., Latiff, M.S.A., Madni, S.H.H., Abdullahi, M.: Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. Neural Comput. Appl. **29**(1), 279–293 (2016)
60. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software **41**(1), 23–50 (2011)
61. Buyya, R., Ranjan, R., Calheiros, R.N.: Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In: High Performance Computing & Simulation, 2009. HPCS'09. International Conference on 2009, pp. 1–11. IEEE
62. HPC2N: The HPC2N Seth log; 2016. http://www.cs.huji.ac.il/labs/parallel/workload/l_hpc2n/
63. NASA: The NASA Ames iPCS/860 log; 2016. http://www.cs.huji.ac.il/labs/parallel/workload/l_nasa_ipsc/
64. Barquet, A.L., Tchernykh, A., Yahyapour, R.: Performance evaluation of infrastructure as service clouds with SLA constraints. Comput. Sist **17**(3), 401–411 (2013)
65. Zhan, J., Wang, L., Li, X., Shi, W., Weng, C., Zhang, W., Zang, X.: Cost-aware cooperative resource provisioning for

heterogeneous workloads in data centers. IEEE Trans. Comput. **62**(11), 2155–2168 (2013)
66. Mehrotra, P., Djomehri, J., Heistand, S., Hood, R., Jin, H., Lazanoff, A., Saini, S., Biswas, R.: Performance evaluation of Amazon Elastic Compute Cloud for NASA high-performance computing applications. Concurr. Comput. **28**(4), 1041–1055 (2013)
67. Tchernykh, A., Lozano, L., Schwiegelshohn, U., Bouvry, P., Pecero, J.E., Nesmachnow, S., Drozdov, A.Y.: Online bi-objective scheduling for IaaS clouds ensuring quality of service. J. Grid Comput. **14**(1), 5–22 (2016)
68. Abdulhamid, S.I.M., Latiff, M.S.A., Abdul-Salaam, G., Madni, S. H.H.: Secure scientific applications scheduling technique for cloud computing environment using global league championship algorithm. PloS ONE **11**(7), e0158102 (2016)
69. Abdullahi, M., Ngadi, M.A.: Symbiotic Organism Search optimization based task scheduling in cloud computing environment. Future Gener. Comput. Syst. **56**, 640–650 (2016)
70. Kruekaew, B., Kimpan, W.: Virtual machine scheduling management on cloud computing using artificial bee colony'. In: Proceedings of the International MultiConference of Engineers and Computer Scientists 2014, pp. 12–14
71. Kimpan, W., Kruekaew, B.: Heuristic task scheduling with artificial bee colony algorithm for virtual machines. In: Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems, 2016 Joint 8th International Conference on 2016, pp. 281–286. IEEE
72. Chen, Z.-G., Du, K.-J., Zhan, Z.-H., Zhang, J.: Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm. In: 2015 IEEE Congress on Evolutionary Computation (CEC) 2015, pp. 708–714. IEEE
73. Kashan, A.H.: League championship algorithm: a new algorithm for numerical function optimization. In: Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of 2009, pp. 43–48. IEEE
74. Eberhart, R.C., Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on 2000, pp. 84–88. IEEE
75. Marichelvam, M., Prabaharan, T., Yang, X.-S.: Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. Appl. Soft Comput. **19**, 93–101 (2014)
76. Ouaarab, A., Ahiod, B., Yang, X.-S.: Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput. Appl. **24**(7–8), 1659–1669 (2014)

Syed Hamid Hussain Madni is a Ph.D. scholar at Faculty of Computing, Universiti Teknologi Malaysia. His area of research is "Resource Management for Infrastructure as a Service in Cloud Computing". He has received M.S. (C.S.) degree in 2009 from Federal Urdu University Arts, Science and Technology, Islamabad, Pakistan. He has completed his B.S. (IT) degree in 2005 from Allama Iqbal Open University, Islamabad, Pakistan. His areas of interest are cloud computing, analysis of algorithm, network security, e-commerce, web development and Internet of Things.

Muhammad Shafie Abd Latiff received his Ph.D. degree in 2000 from Bradford University, United Kingdom. He is an Associate Professor and currently the member of Pervasive Computing Research Group at the Faculty of Computing, Universiti Teknologi Malaysia (UTM). His research interests are in computer networks with focus generally on routing protocol, grid and cloud computing, wireless sensor networks. He successfully supervised 11 Ph. D. students in related fields. Currently, he leads the research project on Scheduling Technique for Resource Provisioning on Cloud and IoT environments. He is the member of IEEE and ACM. At the Malaysian national level he is the committee member of Industry Standards Committee on Information Technology, Communications and Multimedia (ISC G), SIRIM and the committee member of Communication Industrial Standard (CISC), Malaysian Communications and Multimedia Commission. He also active as the Council Member of Cisco Networking Academy Malaysia from 2006 till now.

Shafi'i Muhammad Abdulhamid received his Ph.D. degree in 2016 from Faculty of Computing, Universiti Teknologi Malaysia. He received his M.Sc. degree in Computer Science from Bayero University Kano, Nigeria and B.Tech. degree in Mathematics/Computer Science from the Federal University of Technology Minna, Nigeria. His current research interests are in grid computing and cloud computing. He is a member of Computer Professional Registration Council of Nigeria (CPN), IEEE Computer Society and a member of Nigerian Computer Society (NCS). Presently he is a working as Assistant Professor at Department of Cyber Security Science, Federal University of Technology Minna, Niger State, Nigeria.

Javed Ali is presently working as Assistant Professor in the College of Computing Informatics at Saudi Electronic University, Madinah Munawarah, Kingdom of Saudi Arabia. He received B. Sc.(Hons), M.C.A. and Ph.D. from Aligarh Muslim University, Aligarh. He has 5 years of teaching experience of reputed International and National Universities viz Glocal University, Uttar Pradesh, India, ACN College of Engineering and Technology. His research interest includes parallel and distributed computing, expert systems, cloud computing and Internet of Things. He has published about 14 research papers in Journals/Conferences of International repute. He received state-level scientist award by the government of India.