Cluster Computing
https://doi.org/10.1007/s10586-020-03075-5

A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments

Laith Abualigah[1] · Ali Diabat[2,3]

Received: 24 December 2019 / Revised: 6 February 2020 / Accepted: 17 February 2020

Abstract
Efficient task scheduling is considered as one of the main critical challenges in cloud computing. Task scheduling is an NP-complete problem, so finding the best solution is challenging, particularly for large task sizes. In the cloud computing environment, several tasks may need to be efficiently scheduled on various virtual machines by minimizing makespan and simultaneously maximizing resource utilization. We present a novel hybrid antlion optimization algorithm with elite-based differential evolution for solving multi-objective task scheduling problems in cloud computing environments. In the proposed method, which we refer to as MALO, the multi-objective nature of the problem derives from the need to simultaneously minimize makespan while maximizing resource utilization. The antlion optimization algorithm was enhanced by utilizing elite-based differential evolution as a local search technique to improve its exploitation ability and to avoid getting trapped in local optima. Two experimental series were conducted on synthetic and real trace datasets using the CloudSim tool kit. The results revealed that MALO outperformed other well-known optimization algorithms. MALO converged faster than the other approaches for larger search spaces, making it suitable for large scheduling problems. Finally, the results were analyzed using statistical t-tests, which showed that MALO obtained a significant improvement in the results.

Keywords Task scheduling · Multi-objective optimization · Differential evolution · Virtual machines · Antlion optimization algorithm · Meta-heuristic algorithms · Optimization problem

1 Introduction

To work efficiently with the growing computational need of enormous scale applications, cloud computing (CC) allows a high-velocity deployment of enormous scale applications in recent days, because the cloud gives flexible and resilient computing devices/resources, which can be hired on the pay-per-use model [1]. Massive-scale applications consist of a tremendous number of jobs/tasks, which are performed on the environment as a service cloud. The services in cloud computing are, as shown in Fig. 1, given in the class of software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). SaaS co-operation model passes cloud applications to the users through the Internet and these cloud applications are reached utilizing dependent/client applications such as web browsers on a desktop computer or workstation. It is normally employed for service cloud applications such as web-mail sites, video-sharing sites, social networking sites, and text document editing sites. PaaS presents developing applications with a suitable environment for improvement, examination, and accommodation for their applications [2].

Furthermore, IaaS gives access to scalable and elastic computing devices for spreading wide-scale applications. By the IaaS model, virtualized computing devices called virtual machines (VMs) with pre-configured CPU processor, storage, area, memory, and bandwidth are used by the

✉ Laith Abualigah
laythdyabat@aau.edu.jo

[1] Faculty of Computer Sciences and Informatics, Amman Arab University, Amman, Jordan

[2] Division of Engineering, New York University Abu Dhabi, Saadiyat Island 129188, Abu Dhabi, United Arab Emirates

[3] Department of Civil and Urban Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201, USA

Published online: 12 March 2020

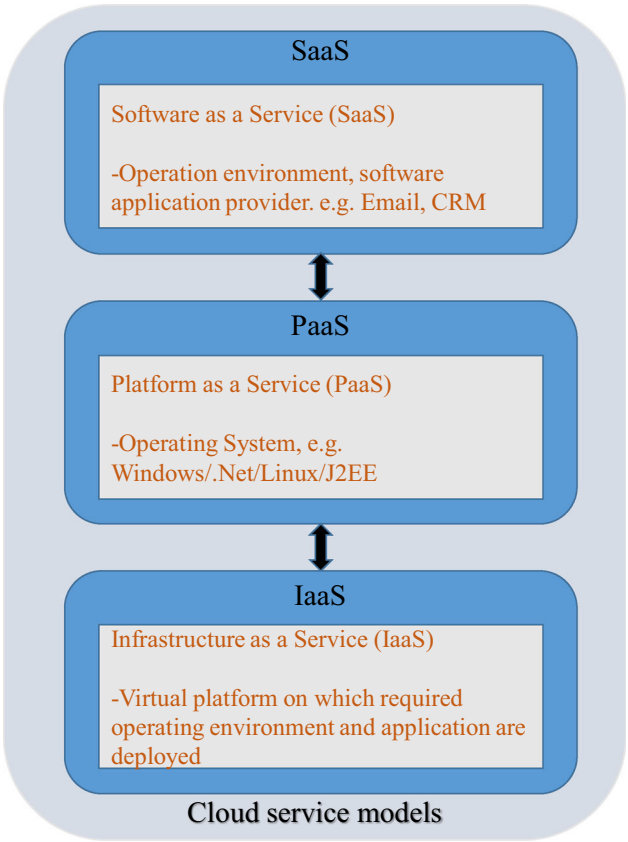


Fig. 1 Kinds of cloud service providers

end users to find what is the suitable use. Many VM situations are accessible to the users at various prices to assist their numerous application needs, and this grants users the right to manage the computer resources at the end. IaaS gives three basic benefits to the end-users. The first benefit is that users employ resources based on needs, and they pay per usage of the infrastructure similar to how one might pay for basic services such as power, gas, and water. This lets users contract or increase their resources agreement based on the demands of their application. The second benefit is that IaaS cloud computing gives straight resources provisioning which enhances the production of user applications. The third benefit, users can acquire rented resources anywhere and anytime based on the coveted level of service. Finding the satisfactory number of resources to perform a collection of massive-scale assignments on IaaS cloud is yet an open and well-known problem [3, 4].

By virtue of the functional applications and difficulties of performing massive-scale requests, task scheduling of applications on the massive scale has fallen under the emerging investigation in the cloud computing environment and has attracted vital attention of scholars in recent days [5, 6]. Several meta-heuristic algorithms have been employed to tackle task scheduling problems and other optimization problems which produce optimal solutions for small volume problems [7–9]. Nevertheless, the quality of the variety of candidate solutions generated by these optimization techniques deteriorates woefully as the problem volume and number of variables options to be optimized increments. Moreover, these techniques do not own supply and assistance for reaching various QoS demands. On the other hand, various cloud users want specific QoS satisfaction, particularly for scientific and industry region applications [10]. Recently, efforts have been produced to solve task scheduling problems using meta-heuristic algorithms such as genetic algorithms (GA), moth search algorithm (MSA), particle swarm optimization (PSO), whale optimization algorithm (WOA), and ant colony optimization (ACO) [11–13]. Employing meta-heuristic techniques for tackling task scheduling problems in cloud computing have revealed promising developments in obtaining effective performance, by decreasing the solution search region [14–16]. However, these techniques result in the high computational running time, and in some instances, yield the local optimum solution, particularly when trading with large solution search regions; moreover, sometime these techniques may suffer from early convergence (stuck/trapped in local search), imbalance between local search and global search strategies, and its exploitation search-ability is not as stable as its exploration search-ability [3, 17].

These limitations affect the obtained task schedule solutions which influence the production (effectiveness) of service offered with regard to reaching the coveted QoS aims. Furthermore, most current works fail miserably to achieve the vital features that are fundamental to CC such as heterogeneity, flexibility, dynamism, and elasticity; therefore, they fail to accomplish the user's QoS demands. Consequently, there is a need for optimization-based meta-heuristic algorithms that can efficiently acclimate with wide-search-space when scheduling massive-scale cloud computing applications. Hence, there is a direction for further improvement in the solutions and further enhanced solutions of the task scheduling problem. Therefore, in this paper, a novel multi-objective optimization using hybrid antlion optimizer (ALO) with elite-based differential evolution (DE) for task scheduling problems in cloud computing environments is presented (MALO).

The exploitation capability of Antlion optimizer (ALO) still needs to be enhanced; therefore, the elite-based differential evolution can be utilized as a local search technique. This paper introduced a novel multi-objective optimization method using hybrid antlion optimizer algorithm (MALO) for task scheduling problems in cloud computing environments with balanced task configuration/distribution. The introduced algorithm is hybridized with a

local search technique, differential evolution (DE) strategy, to improve the exploitation search-ability of the ALO since the DE algorithm demonstrates the powerful features of the genetic algorithm (GA) and the evolution strategy (ES).

These features are the extended population of the genetic algorithm using the crossover operator and self-adapting mutation of evolution strategy. Based on the characteristics, it has been confirmed that it enhanced the effectiveness of other meta-heuristic algorithms. Example, Zheng et al. [18] applied the differential evolution strategy to improve the effectiveness of fireworks algorithm to distribute the data between the fireworks and sparkles. Also, Yazdi et al. [19], mixed the differential evolution strategy with harmony search (HS) algorithm and employed it as a multi-objective scheme of water distribution networks. Yuancheng et al. [20] introduced a hybrid chaotic artificial bee colony algorithm with differential evolution strategy for the problem of reactive energy optimization. The firefly optimization algorithm is developed, by Zhang et al., as a global search technique by combining with the differential evolution strategy as performed in [21].

The main contributions of this paper are:

- Design of a discrete ALO algorithm for solving the tasks' scheduling problem in a cloud computing environment.
- Present an alternative meta-heuristic technique based on the hybrid antlion optimizer (ALO) algorithm with the differential evolution (DE) strategy (MALO) for solving task scheduling problems in the cloud computing environment using synthetic and real trace data.
- Present a multi-objective optimization based on the proposed MALO to decrease the makespan value and to enhance the resource utilization together.
- Use a statistical test (i.e., *t*-test) to validate the obtained results of the proposed MALO against the well-known comparative methods using a significance test.

The remaining sections of this paper are organized as follows. Section 2 presents the related works that have been used to solve the task scheduling problems. Section 3 explains the basic concepts and definitions of tasks and machines as well as the proposed task scheduling framework. Section 4 presents the general procedures of the proposed multi-objective hybrid Antlion Optimizer for task scheduling. Finally, the experimental plan, performance evaluation, and results analysis of the proposed multi-objective hybrid antlion optimizer algorithm (MALO) are reported in Sect. 5.

2 Related works

This section presents related works that have been used in the literature to solve the task scheduling problems using optimization methods. The task scheduling in cloud computing environments is NP-hard complete problem, so finding an accurate solution (optimal) is ungainly, particularly for big task sizes. Several methods in the literature have been proposed to solve this problem [22–27].

An effective binary variant of PSO algorithm with low cost and time complexity is proposed in [28] for addressing the scheduling and balancing tasks problem in cloud computing. Particularly, an objective function to measure the maximum computational time among different virtual machines is defined to optimize this problem. Finally, a particle updating is devised to maintain load balancing. In [29], a new method using modified PSO algorithm (M-PSO) is proposed to address the task scheduling problem and to deal with the local optimum and premature convergence problem. Distinct from the basic PSO, the proposed M-PSO can dynamically change the inertia weight value to enhance the convergence speed based on the number of generations. The obtained results proved that the proposed M-PSO can decrease total cost compared with other comparative methods. Yassa et al. [30] solved the scheduling workflow problem on various computing schemes such as cloud computing foundations. This work introduced a novel multi-objective optimization approach for workflow scheduling problem in clouds and introduced the hybrid PSO algorithm to determine the optimal scheduling performance. The obtained results highlight the robust performance of the proposed multi-objective optimization approach. Ben Alla et al. [31] introduced a new method to solve the task scheduling using a new structure with Dynamic Queues based on a hybrid PSO algorithm using Fuzzy Logic (TSDQ-FLPSO). The proposed algorithm aimed at finding the optimal makespan and expecting time. The obtained results based on the simulator (CloudSim) showed that the proposed method (TSDQ-FLPSO) performed the optimal stability results, reducing the waiting time, decreasing the makespan value, and enhancing the resources utilization compared to other comparative algorithms. Several related works are in [32].

The basic version of this algorithm, symbiotic organism search (SOS), is recently introduced as an optimization technique for addressing mathematical optimization problems. A discrete symbiotic organism search (DSOS) algorithm is proposed in [33] for finding the scheduling of tasks on cloud devices. The obtained results showed that DSOS exceeds the results of PSO for solving the task scheduling problems. DSOS converges more quickly when the search grows larger which gives it suitable behavior for big-scale

problems. In [3], a chaotic SOS (CMSOS) algorithm is introduced to determine multi-objective large task scheduling problem on IaaS cloud computing ecosystem. A chaotic strategy is applied to produce the initial population, and stochastic sequence-based parts in SOS are replaced with chaotic distribution to guarantee heterogeneity among organisms for fast convergence. Therefore, the proposed CMSOS improved the QoS performance compared with other methods.

An alternative approach for the problem of cloud task scheduling is proposed in [34]. This problem tries to reduce makespan value that is needed to schedule multiple tasks on several Virtual Machines. The introduced method is based on using the hybrid moth search algorithm (MSA) with the differential evolution (DE) strategy. But, the exploitation ability of MSA needs to be enhanced; hence, the differential evolution can be utilized as a local search technique. To evaluate the performance of the proposed MSDE algorithm, sets of three experimental series are performed. Two experimental series are carried out and the results prove that the proposed MSA got better results compared to the other algorithms according to the effectiveness measures.

For solving the task scheduling problems, a multi-objective optimization method is introduced in [35]. A multi-objective scheduling scheme is introduced with an enhanced ant colony optimization (ACO) algorithm to address this problem. The obtained results show that the proposed method achieved better results than other comparable methods, particularly as it improved by 56.6% in the best-case situation. In [36], a new algorithm to solve cloud task scheduling problem is based on the ACO algorithm that assigns tasks to virtual machines in the cloud computing ecosystem in an effective way. To improve the production of the task scheduler with the ACO algorithm, diversification and reinforcement approaches are adapted. The proposed algorithm got better results in solving that problem effectively compared with other similar methods.

Agarwal and Srivastava [37] introduced a novel meta-heuristic method based on the genetic algorithm (GA) and PSO (PSOGA). The proposed algorithm (PSOGA) utilizes the diversification part of PSO and intensification part of the GA. The results of the proposed method showed its ability compared with other methods. Nzanywayingoma and Yang [38] introduced a hybrid method using GA and PSO to solve the task scheduling problem. The results are tested using benchmark test functions and the results revealed that the proposed hybrid method exceeds the results of the original PSO and reduces the execution time. Zheng and Wang [39] introduced a Pareto method using fruit fly optimization (FFO) algorithm to determine the task scheduling and resources management problem in the cloud computing ecosystem (PFOA). The non-controlled sorting system using the Pareto optimum is utilized and observed memory is applied to deal with various objectives in addressing that problem by the proposed method (PFOA). The obtained results showed that the proposed PFOA exhibited competitive results compared with other methods. Several related works are in [40–45]. An overview of the related works are given in Table 1.

3 Preliminaries

This section explains the basic concepts and definitions of tasks and machines as well as the proposed task scheduling framework.

3.1 Task scheduling problem

The problem of task scheduling in the cloud is defined as how to schedule, distribute, and assign many different tasks to many virtual machines effectively and to perform all the tasks to be accomplished in low execution time [4, 46, 47]. The cloud system (CS) includes (N_{pm}) physical machines (PM), and each machine includes (N_{vm}) virtual machines as shown in Eq. 1).

$$CS = [PM_1, PM_2, \ldots, PM_i, \ldots, PM_{N_{pm}}] \tag{1}$$

where PM_i, (i = 1, 2,…, N_{pm}) denotes the physical machines performed in the cloud and it can be expressed as follows:

$$PM = [VM_1, VM_2, \ldots, VM_k, \ldots, VM_{N_{vm}}] \tag{2}$$

where VM_k, (k = 1, 2,…, N_{vm}) denotes the k_{th} virtual machine. N_{vm} denotes the number of virtual machines and VM_k denotes the k_{th} virtual machine device in the cloud. The feature of VM_k is determined as follows:

$$VM_k = [SIDV_k, mips_k] \tag{3}$$

where id denotes the identifier number of virtual machines and $MIPS_k$ denotes the report processing acceleration of virtual machines by millions-of-instructions-per-second [48, 49].

$$T = [Task_1, Task_1, \ldots, Task_i, \ldots, Task_{N_{tsk}}] \tag{4}$$

where N_{tsk} denotes the number of tasks i presented by the users. $Task_i$ denotes the i_{th} task in the tasks series that is determined as follows:

$$Task_i = [SIDT_i, Task - length_i, ECT_i, LI_i] \tag{5}$$

where $SIDT_i$ denotes the identity number of the i_{th} task and $task - length_i$ denotes length of the task. Time ECT_i denotes the foreseen completion time for the i_{th} task; LI_i denotes the task preference in the number of tasks N_{tsk}

Table 1 An overview of the related works

Algorithm	Proposed	Contributions	Measures	References
PSO	Binary version of PSO	A particle updating is devised to maintain load balancing	Computational time	[28]
PSO	Modified PSO algorithm (M-PSO)	Dynamically change the inertia weight value	Total cost	[29]
PSO	Hybrid PSO algorithm	Introduced the hybrid PSO algorithm to determine the optimal scheduling performance	Performance	[30]
PSO	Hybrid PSO algorithm	A new structure with Dynamic Queues based on a hybrid PSO algorithm using Fuzzy Logic (TSDQ-FLPSO)	Makespan value	[31]
SOS	A discrete symbiotic organism search (DSOS) algorithm	A discrete symbiotic organism search (DSOS) algorithm is proposed	Makespan value	[33]
SOS	A chaotic SOS	Achaotic SOS (CMSOS) algorithm is introduced to determine multi-objective large task scheduling problem	Convergence	[3]
MSA	A hybrid MSA	The hybrid moth search algorithm (MSA) with the differential evolution (DE) strategy	Makespan value	[34]
ACA	An enhanced ACA	A multi-objective scheduling scheme is introduced with an enhanced ant colony optimization (ACO)	Makespan value	[35]
ACO	An improved ACO	A new algorithm to solve cloud task scheduling problem is based on the ACO algorithm	Makespan value	[36]
GA-PSO	A hybrid genetic algorithm (GA) and PSO (PSOGA)	A novel meta-heuristic method based on the genetic algorithm (GA) and PSO (PSOGA). The proposed algorithm (PSOGA) utilizes the diversification part of PSO and intensification part of the GA	Makespan value	[37]
FFO	A Pareto method using FFO algorithm	A Pareto method using fruit fly optimization (FFO) algorithm to determine the task scheduling and resources management problem in the cloud computing ecosystem (PFOA)	Makespan value	[39]

[50, 51]. The Expected Complete Time (ECT) measure of size $N_{tsk} \times N_{vm}$ denotes the execution time needed to perform the task on each computing device (virtual machine) that can be determined by the following matrix (3.1). An example of the measure of the expected time to compute is given in Table (2):

$$ECT = \begin{bmatrix} ECT_{1,1} & ECT_{1,2} & ECT_{1,3} & \cdots & ECT_{1,N_{vm}} \\ ECT_{2,1} & ECT_{2,2} & ECT_{2,3} & \cdots & ECT_{2,N_{vm}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ ECT_{N_{tsk},1} & ECT_{N_{tsk},2} & \cdots & \cdots & ECT_{N_{tsk},N_{vm}} \end{bmatrix}$$

3.2 Task scheduling problem definitions

Definition 1 (*Virtual machines* (*VMs*)) Usually, each virtual machine can be represented as a tuple/row ($VM = \{id, mips, bw, pes - number\}$), where *id* denotes to the identifier number of a virtual machine, and *mips* means a million instructions per second. *mips* denotes to the average execution time for each processing element (*PE*) of each virtual machine, *bw* denotes to the bandwidth of a virtual machine and $pes - number$ denotes to the number of processing elements in each virtual machine [28, 52, 53].

Definition 2 (*Tasks/jobs to be scheduled* (*Ts*)) A task (T) can be defined as tuple/row ($T = \{id, length, pes - number\}$), where *id* here is the identifier number of the task *T*, *length* denotes to the size of *T* in million instructions (*MI*) and $pes - number$ denotes to the number of processing elements for executing the assigned task on the proper virtual machine.

Definition 3 (*Near-optimal solution*) In this paper, the near-optimal solution is described as the assignment of a diverse set of given tasks into several different virtual machines that seek to minimize the makespan time, waiting for time, and level of imbalance. At the same time, the near-optimal solution will maximize device exploitation and will minimize execution time and cost.

Definition 4 (*Degree of imbalance* (*DI*)) Degree of imbalance is an evaluation measure to test the volume of load distribution over the virtual machines in terms of their performance and execution competencies. The small value of the level of imbalance means that the load of the distribution process is more stable (balanced). Degree of imbalance is determined by Eq. (6)

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \tag{6}$$

where T_{max} denotes to the maximum execution time achieved, T_{min} denotes to the minimum execution time achieved, T_{avg} denotes to the average complete execution time achieved through all the virtual machines.

Definition 5 (*Completion time* (*CT*) *of the virtual machine*) Completion time of the i_{th} virtual machine is denoted as the running time after executing the last task on i_{th} virtual machine (CT_i) [54, 55]. It is determined by Eq. (7)

$$CT_i = \sum_{j=1}^{n} \frac{T_{i.length}}{VM_{i.pesnumber} \times VM_{i.mips}}, \tag{7}$$

where, i denotes the number of virtual machines and the i value is within $\{1, 2, 3, \ldots, m\}$, denotes the number of tasks/jobs, and the j value is within $\{1, 2, 3, \ldots, n\}$.

Definition 6 (*Makespan*) Makespan is the overall accomplishment time required to complete the execution of all tasks. On the other hand, in terms of manufacturing, **makespan** is the time interval between the start point and finish point of a sequence of jobs/tasks. The makespan means that if its value is low, the scheduler is giving ideal and effective planning steps of tasks to devices (virtual machine). And if the value of the makespan is high, the scheduler is not giving ideal and effective planning steps of tasks to devices. It is determined by Eq. (8)

$$makespan = \max_{1 \le i \le m} \{CT_i\} \tag{8}$$

Definition 7 (*Resource utilization* (*Ru*)) Resource utilization is a performance measure to compute the utilization of devices/resources. A high utilization price/value means that the cloud provider gets the maximum profit. It is determined by Eq. (9)

$$Ru = \frac{\sum_{i=1}^{m} CT_i}{makespan \times m} \tag{9}$$

Definition 8 (*Execution cost* (*EC*)) Execution cost denotes the cost of the cloud computing user to cloud computing provider against the exploitation of devices to perform tasks. The main objective for cloud computing users is to decrease the cost alongside with effective utilization and smallest makespan. It is determined by Eq. (10)

$$EC = \sum_{i=1}^{m} price_i * CT_i \tag{10}$$

3.3 Mapping antlion optimizer for task scheduling

To implement the proposed MALO to solve the task scheduling problem, the ant is defined as a multi-dimensional matrix in which m x n positions exist. This represents a candidate solution, which presents the distribution of tasks into various virtual machines [56]. Referring to the given matrix in Table 2, each column denotes a task position and each row denotes earmarked tasks to a virtual machine. In each line, 1 is utilized to represent that a virtual machine is specified to a task and each task will be performed by one virtual machine only while 0 shows non-assignment task [57, 58].

For illustration, Table 3 shows modeling of eight tasks on four virtual machines. Similar to the position matrix, each ant value is also expressed in the structure of $m \times n$ matrices and the area for its components is [0, 1]. Similar to Table 2, the best position of the ant and the global exists in m x n matrices. 0 and 1 denotes a matrix that gives the best-obtained distribution of tasks into different virtual machines. The ants and the global show the best distribution of tasks into heterogeneous different virtual machines [59, 60].

Consequently, the tasks scheduling problem in this paper can be expressed as two sub-problems as follows:

1. How to obtain the optimal schedule and balance different tasks to heterogeneous virtual machines in cloud computing using the proposed antlion optimizer algorithm with low cost.
2. How to obtain low time complexity of the proposed antlion optimizer algorithm to make it beneficial in practical situations.

3.4 The proposed multi-objective task scheduling function

The most important objective function is decrease the makespan value (in Eq. 11) by arranging the most suitable set of tasks to be performed on virtual machines. Also, the resource utilization (in Eq. 12) is considered alongside with makespan value, which is a performance measure to

Table 2 An example of the measure of the expected time to compute

	T1	T2	T3	T4	T5
V1	T1/V1	T2/V1	T3/V1	T4/V1	T5/V1
V2	T1/V2	T2/V2	T3/V2	T4/V2	T5/V2
V3	T1/V3	T2/V3	T3/V3	T4/V3	T5/V3
V4	T1/V4	T2/V4	T3/V4	T4/V4	T5/V4
V5	T1/V5	T2/V5	T3/V5	T4/V5	T5/V5

Table 3 An example of the ant k for eight independent tasks and four virtual machines

	Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8
VM1	1	0	0	0	0	0	1	0
VM2	0	1	1	0	0	0	0	1
VM3	0	0	0	0	1	1	0	0
VM4	0	0	0	1	0	0	1	0

compute the utilization of devices/resources. A high utilization price/value means that the cloud provider gets the maximum profit.

$$ECT_{ik} = \frac{task - length_i}{mips_k} \tag{11}$$

$$Ru = \frac{\sum_{i=1}^{m} CT_i}{makespan \times m} \tag{12}$$

where k = 1, 2, 3,…, N_{vm}, i = 1, 2, 3,…, N_{tsk}, and ECT_{ik} denotes the needed execution time of i_{th} task on k_{th} virtual machine. N_{vm} denotes the number of virtual machines and N_{tsk} is the total number of tasks. The combined fitness value with the multi-objective function of each collection can be calculated using Eq. (13), which defines the evolution strength of the organism to the ecosystem [33].

$$F = (max\{ECT_{ik}\}\&min\{Ru_k\}), \quad \forall \in [1, N_{tsk}]\, mapped\, to\, k_{th} VM, k = 1, 2, \ldots, N_{vm} \tag{13}$$

Normally, the requested tasks are scheduled on the free virtual machines and these tasks are served based on its ordering (first-come-first-serve). The main aim of tasks scheduling over the virtual machines is how to obtain higher utilization of virtual machines alongside with lower makespan value. Expected Time to Compute (ETC) of the given tasks to be listed on each virtual machine will be utilized by the proposed algorithm to perform schedule arrangement. Expected Time to Compute values are defined by using the ratio of million-instructions-per-second (MIPS) of a virtual machine to the length of task [61, 62] as shown in Table 2.

As mentioned in [34, 63, 64], this is a multi-objective problem that is hard to solve; in particular, it is hard to get the near-optimal solution and it needs a new method to deal with multi-objective function to solve this problem effectively. Thus, the proposed multi-objective tasks scheduling function is incorporated within the proposed hybrid antlion optimizer algorithm and employed in this paper to address the problem of tasks scheduling in the cloud computing ecosystem. The main aim of the proposed multi-objective function is decreasing the makespan value and enhancing the resources utilization together.

3.5 Preliminaries factors

ALO is a novel algorithm meta-heuristic introduced by Seyedali Mirjalili [65]; it acquired strong considerable attention and interest of scholars from real computing. ALO was first introduced to address constrained problems (i.e., benchmark and engineering problems) with different wide search-spaces. The ALO algorithm also revealed robust effectiveness and more active convergence speed (quicker) when matched with the states of matter search (SMS) optimization algorithm [66], bat algorithm (BA) [67], genetic algorithm (GA) [68], flower pollination (FP) optimization algorithm [69], cuckoo search (CS) optimization algorithm [70], particle swarm optimization (PSO) algorithm [71], and firefly optimization algorithm (FA) [72], which are the popular common meta-heuristic algorithms.

ALO has determined to be an efficient and useful algorithm for addressing complex optimization multi-dimensional problems with wide search-space while controlling multi-objective optimization problems and constrained optimization problems. Active scholars who worked on the ALO algorithm since the time of its launch have studied modification versions, discrete optimization problems, constrained, binary versions, multi-objective optimization problems, and hybridization versions. Hybridization is designed to combine the powers and strengths of ALO such as global search-ability and accelerated optimization with other components from well-known similar optimization techniques to solve some of the issues of weakness associated with ALO performance, such as getting trapped in local optima.

4 The proposed antlion optimizer

Antlion Optimizer (ALO) was proposed by Mirjalili in 2015. This algorithm mimics the way that antlions dig sand pits to hunt ants in nature [65]. The Pseudo-code of the proposed multi-objective hybrid antlion optimizer (ALO) with elite-based differential evolution (DE)(MALO) is presented in Algorithm 1.

In the proposed algorithm (MALO) as seen in Fig. 2, a population of solutions is first initialized as ants on a search

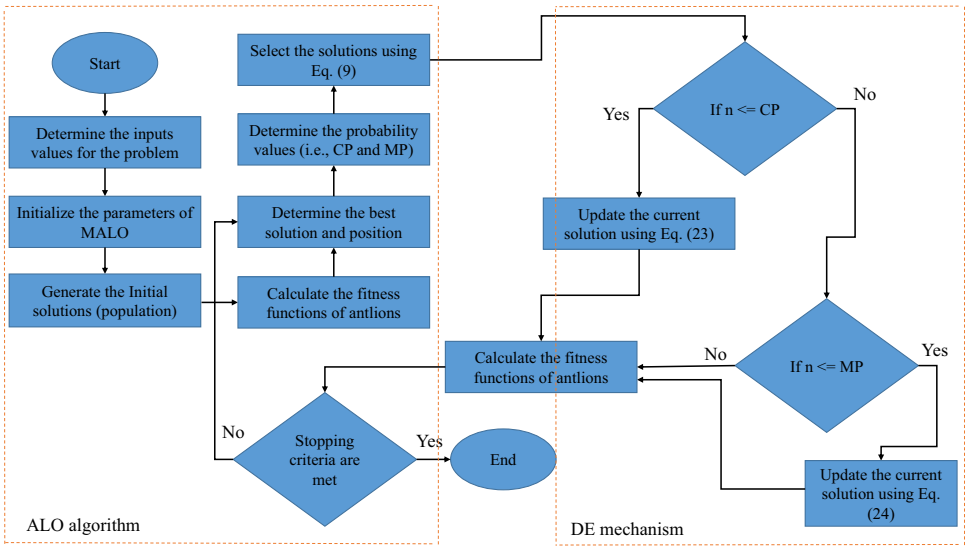


Fig. 2 The proposed MALO method for solving task scheduling

landscape. The position of each ant is stored in a vector as follows:

Algorithm 1 : The procedure of the Ant Lion Optimizer Algorithm

```
Initialize the random solutions (i.e., ants and antlions)
Calculate the fitness function (i.e., ants and antlions)
Find the best antlions and assume it as the optimal so far

while The termination criterion is not reached do
    for each solution (ant) do
        Select an antlion using Roulette wheel
        Update the perimeters c and d
        Create a random walk
        normalize the chosen random walk

        if n¡=CP then
            Update the current solution by using Eq. (15)
        else if n¿CP then
            Update the current solution by using Eq. (23)
        end if

        if n¡=CP then
            Update the current solution by using Eq. (23)
        end if

        Calculate the fitness function of all solutions using Eq. (13)
        Replace an antlion (new solution) with its corresponding ant (current) if becomes fitter
    end for
    Update the current best solution if an antlion becomes fitter than the old best
end while

return Thebestsolution(elite)
```

$$\overrightarrow{Ant_i} = \left[A_{i,1}, A_{i,1}, \ldots, A_{i,d}\right] \tag{14}$$

where Ant_i shows i^{th} ant, $A_{i,d}$ shows the position of the i^{th} ant in the d^{th} dimension.

The position of each ant in each dimension is updated using a random walk. This random walk is as follows:

$$x(t) = [0, \text{cumsum}(2t(t_1)) - 1, \text{cumsum}(2t(t_2)) - 1, \ldots, \text{cumsum}(2t(t_T)) - 1] \tag{15}$$

where T is the maximum number of iterations, t_i shows the t^{th} iteration, cumsum is the cumulative summation, and $r(t)$ is a random function calculated as follows:

$$r(t) = \begin{cases} 1 & rand \geq 0.5 \\ 0 & rand < 0.5 \end{cases} \tag{16}$$

where t indicates is the iteration index and *rand* is a randomly generated number in [0, 1]

To see how this random walk works, Fig. 3 is provided. This figure shows 50 random walks using the same equation. It can be seen that the deviations can be quite abrupt starting from the first iteration.

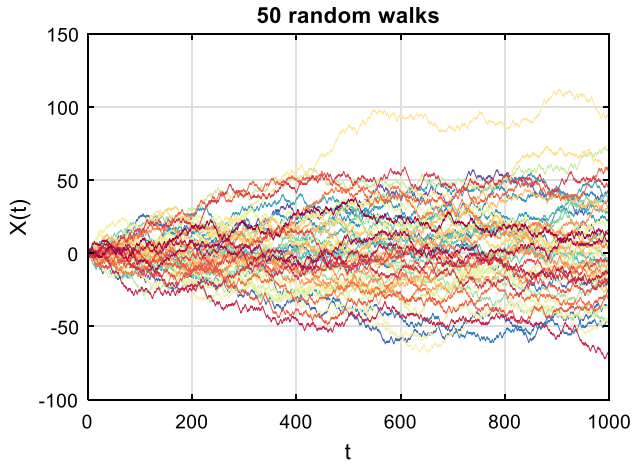


Fig. 3 Fifty random walks using Eq. 15

ALO is population-based algorithm [73, 74], so multiple ants are stored in a population matrix as follows:

$$M_{ant} = \begin{bmatrix} \overrightarrow{Ant_1} \\ \overrightarrow{Ant_2} \\ . \\ . \\ \overrightarrow{Ant_n} \end{bmatrix} \tag{17}$$

where n is the number of ants in the population.

In ALO, each ant is also evaluated using an objective function [75]. Therefore, we need a vector to store the result as follows:

$$M_{oa} = \begin{bmatrix} f(\overrightarrow{Ant_1}) \\ f(\overrightarrow{Ant_2}) \\ . \\ . \\ f(\overrightarrow{Ant_n}) \end{bmatrix} \tag{18}$$

With the above equation the positions and objective of ants can be calculated as follows. Each ant represents a solution for a given optimization problem. However, they have to be guided towards the promising regions of the search space. This is where the antlions come into play.

Each antlion is also represented with a position vector and objective vector as follows:

$$\overrightarrow{Antlion_i} = \left[A_{i,1}, A_{i,1}, ..., A_{i,d}\right] \tag{19}$$

where $Antlion_i$ shows i^{th} antlion, $A_{i,d}$ shows the position of the i^{th} ant in the d^{th} dimension.

$$M_{Antlion} = \begin{bmatrix} \overrightarrow{Antlion_1} \\ \overrightarrow{Antlion_2} \\ . \\ . \\ \overrightarrow{Antlion_n} \end{bmatrix} \tag{20}$$

$$M_{oal} = \begin{bmatrix} f(\overrightarrow{Antlion_1}) \\ f(\overrightarrow{Antlion_2}) \\ . \\ . \\ f(\overrightarrow{Antlion_n}) \end{bmatrix} \tag{21}$$

where n is the number of ants in the population.

The ALO moves each ant in a search landscape using the random walk as discussed above. Each variable in each ant faces a different random walk, which increases the exploration of this algorithm. Random walks should be normalized with respect to the upper and lower bounds of each variable. These random walks should be gravitated towards antlions, which simulates how ants get trapped in antlion sand pits in nature. Fitter antlions have a higher probability of impacting ant movements. This is simulated using a roulette wheel. An ant's movement can be impacted by only two ants: one selected using a probability and the best antlion, which is called the elite. The range of all random walks decreases proportionally to the number of iterations. Antlions get updated if we find an ant with better fitness.

To see the behavior of ALO when solving optimization problems, several subplots are given in Fig. 4.

The first column in this figure shows that the first test function is a unimodal one and the second one is a multimodal test function. The second column shows the history of positions that ants and antlions visited during the optimization process. High exploration is evident in these subplots. The third column shows that the range of random walks decreases proportionally to the number of iterations. This causes transitions from exploration to exploitation. The fourth column shows that changes that an ant faces in one of its dimensions. It can be seen that the ant faces abrupt changes in the exploration phase. In the exploitation phase, the changes decrease substantially since the focus is on the improvement of the solutions. A similar pattern can be seen in the fifth column, in which the average fitness of all ants and antlions decreases over the course of iteration. This shows that the mechanisms of ALO improve the first randomly generated population of ants. Finally, the last column shows that the elite antlions also improve, which is also an indication of the convergence of the ALO algorithm.

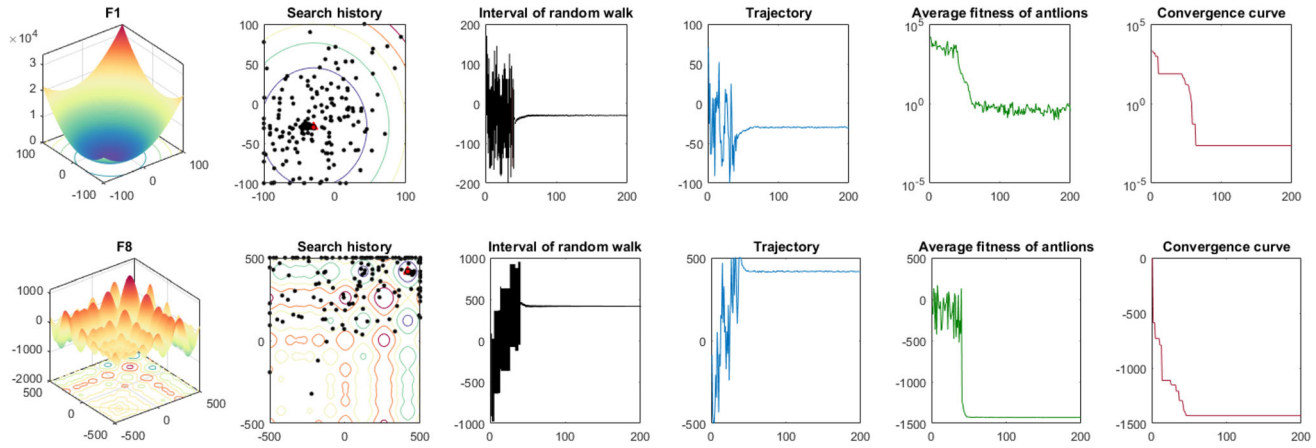


Fig. 4 Performance observation of ALO on two optimization test functions

4.1 Differential evolution

This section shows the main theories of differential evolution mechanism [76]. Generally, this mechanism utilized three main operators, selection, mutation, and crossover, to develop current solutions (population) [18, 19]. Suppose that the population M with size n is created and the procedures of updating its solutions using the mentioned three operators (more details are in Fig. 2), can be explained as the following points [34, 57]:

The selection mechanism is employed to choose the best current solution from the produced solutions and the current solution according to its fitness function value to move to the next iteration. This mechanism is determined as in Equation (22):

$$v_{(t+1),i} = \begin{cases} y_{t,ij} & \text{if} F_{(y)} \leqslant F_{(x_{t,i})} \\ x_{t,i} & \text{otherwise} \end{cases} \tag{22}$$

where F_y is the fitness value of the suggested solution y.

The crossover (Cr) operator is utilized to create a new solution $y_{t,ij}$ ij, $j = 1, 2, \ldots, N_{tsk}$ according to the new solution, and the current solution $x_{t,ij}$ using Eq. (23):

$$v_{(t+1),i} = \begin{cases} y_{t,ij} & \text{if } n \leqslant CP \\ y_{t,ij} & \text{otherwise} \end{cases} \tag{23}$$

where the $CP \in [0, 1]$ is the crossover probability value, and $n \in [0, 1]$ is a random number.

The mutation (Mu) operator is utilized to produce a mutant solution by mixing a stochastic solution with the interval between two other stochastic solutions according to the probability value (MP), if $n \leqslant CP$, as in Eq. (24):

$$v_{t,i} = x_{t,r1} + \gamma \times (x_{t,r1} - x_{t,r1}) \tag{24}$$

where γ denotes the scaling of row ri, $i = 1, 2, 3$ are respectively independent stochastic integers in [1, N], t is the number of the current iteration. Note, we added the searches mechanism (differential evolution: selection, crossover, and mutation operators) in the proposed MALO to improve the ALO ability in solving the problems of task scheduling in the cloud computing environment. This mechanism played the main role in getting the balance between the exploration and exploitation search strategies and it keeps the diversity of the solutions, which helps the algorithm to avoid the trapped in local search.

4.2 Complexity analysis

In this section, the computational complexity of the proposed MALO algorithms is analysed. The complexity of the MALO is depended on the complication of the ALO, Differential Evolution (DE), and multi-objective (MO). Consequently, the complexity of the proposed MALO method is given as follows:

$$\mathrm{O(MALO)} = X_s\mathrm{O(ALO)} + \mathrm{O(DE)} + \mathrm{O(MO)}$$

where,

$$\begin{aligned} \mathrm{O(ALO)} &= \mathrm{O(t(Dim} \times \mathrm{X} + \mathrm{F} \times \mathrm{X} + \mathrm{X}log\mathrm{X})) \\ \mathrm{O(DE)} &= \mathrm{O(t(Dim} \times \mathrm{X} + \mathrm{F} \times \mathrm{X})) \\ \mathrm{O(MO)} &= \mathrm{O(t} \times \mathrm{X}) \end{aligned}$$

where, t denotes to the total number of iterations, Dim meant to the number of given variables, F signified to the value of fitness function, and X indicated the number of solutions (i.e., population size) updated using ALO.

5 Experiments results: evaluation and discussion

In this section, the experimental plan, performance evaluation, and result analysis and discussion of the proposed multi-objective hybrid antlion optimizer algorithm (MALO) are reported.

5.1 Experiments setting

The effectiveness (performance) and scalability of the proposed task scheduling method using MALO is evaluated and compared with other well-known published methods in the literature. As the proposed MALO method depends on the enhanced basic antlion optimizer using the differential evaluation algorithm, so it is essential to evaluate its production as a global search method (i.e., as a task scheduling solver). According to the characteristics of the introduced MALO, that benefits from the multi-objective function, basic antlion optimizer, and differential evaluation algorithm, it is supposed that the convergence speed of the introduced MALO is stabler than the basic algorithm with a single objective function. Further, the introduced MALO algorithm as a task scheduling solver can produce more favorable outcomes than the other popular and, also, the other meta-heuristic optimization algorithms as the task scheduling standards.

In order to validate the effectiveness of the proposed method (MALO), collections of experiments series are produced in two parts. The first part of the experimental, involved in Sect. 5.2, is done to address the task scheduling problem using synthetic datasets. The second part of the experimental, involved in Sect. 5.3, is done to address the task scheduling problem using real trace datasets.

5.2 Experiments part 1: Evaluation results of synthetic datasets

Investigating new procedures or approaches in the real cloud computing ecosystem, such as Amazon (i.e., EC2) and Microsoft (i.e., Azure), is usually constrained by hardness foundations, such as protection, security, speed and the large cost of money if experiments are reproduced. So, it is hard to conduct such investigations in repeatable, dependable, and scalable ecosystems (environments) using real-world cloud environments [77]. In this paper, experiments are conducted to validate the performance of the proposed MALO method; they were executed in CloudSim infrastructures, which is a toolkit for mimicking Cloud computing scenarios.

Two data-centers were produced, each one holding two hosts. Each host has twenty GB RAM (one host is a dual-core machine and the other is a quadcore machine) and one TB memory storage. Each host has the collective processing power of one million MIPS. Several virtual machines were designed with different generated distributions as 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 and 2000 instances. The more considerable instances give insight into the scalability of production of the introduced algorithms (MALO) with extended problem sizes. The settings of the adjustment parameters for MALO and other comparative algorithm are obtained from (Genetic Algorithm (GA) [78], Discrete Symbiotic Organism Search (DSOS) Algorithm [33], Hybrid Moth Search Algorithm (MSDE) [34], Particle Swarm optimization (PSO) Algorithm [79], Whale Optimization Algorithm (WOA) [80], Moth Search Algorithm (MSA) [81], and Antlion Optimizer (ALO) Algorithm [65]). For the ALO parameters setting, refer to Table 4. The CloudSim test settings are provided in Table 5.

The degree of imbalance of MALO performance compared with other well-knowing algorithms are summarized in Table 6. It is observed that at 100 tasks, the degree of imbalance between the proposed MALO and GA, DSOS, MSDE, PSO, WOA, MSA, and ALO is around 0.9301, 1.6250, 1.7500, 0.8649, 1.2773, 1.0857, 1.239, 1.3253, respectively. Meanwhile, in the enormous size of tasks at 1000 tasks, it achieves the results as 0.9701, 2.4416, 1.8654, 1.0010, 0.9985, 1.0865, 0.9874, 0.9996 for the proposed MALO and GA, DSOS, MSDE, PSO, WOA, MSA, and ALO, respectively. Generally, we can conclude that most of the obtained results were the best using the proposed algorithm (MALO), because it got the smallest degree of imbalance compared with other comparative methods.

The proposed algorithm (MALO) acquires slight improvements according to the makespan measure compared to the other competitive optimization algorithms in a various number of tasks as shown in Table 7. Additionally, at task 100, the makespan values of MALO compared with the original ALO are $-$ 5.56, $+$ 5.06, $+$ 7.89 for the Best, Worst, and Avg makespan, respectively. The proposed MALO at case one (i.e., 100 tasks) got values of improvement in Worst and Avg situations. According to the biggest number of tasks (i.e., 1000 tasks), the makespan values of MALO compared with the original ALO are $+$ 1.10, $+$ 0.43, $+$ 2.28, for the Best, Worst, and Avg makespan, respectively. The proposed algorithm (MALO) reduced the makespan value over all the tasks cases. We concluded that the makespan value is slowly grown with rising the size of the tasks. The average value of makespan when using the modified optimization algorithms is better

Table 4 Parameter settings for the ALO algorithm

Algorithm	Parameter	Value
ALO	*p*-value	Less than 0.05
	w	2 to 9 exponential iteratively
	Number of iterations	1000
	Number of solutions	50

Table 5 The CloudSim test settings

Element	Parameter	Values
Data-center	No. of data-center	2
Cloudlet	No. of cloudlets	100–1000
	Length	1000–2000
Virtual machine	RAM	512 MB
	MIPS	100–1000
	Size	10000
	Bandwidth	1000
	Policy type	Time Shared
	No. of CPUs	1
Host	No. of Hosts	2
	RAM	2048 MB
	Storage	1 million
	Bandwidth	10000

than the traditional optimization algorithms. Meanwhile, the average makespan of the proposed algorithm (MALO) is smaller than the other comparative methods (i.e., GA, DSOS, MSDE, PSO, WOA, MSA, and ALO). This shows better performance of the MALO.

In this paper, the statistical analysis, *t*-test, is conducted according to the values of the makespan measure as shown in Table 7. The one-sided *t*-test is conducted to check whether the makespan measure values achieved by the proposed MALO is significantly less than that of ALO for all task cases using the same termination criteria. This measure is one of the main tests used to measure the effectiveness of schedules. The results indicated that nine out of ten cases (100, 200, 300, 400, 600, 700, 800, 900, and 1000 tasks) showed significant improvement in makespan value between both algorithms (the proposed ALO and original ALO), which means that there is a significant difference between the performance of the pro-

Table 6 The comparison among the task scheduling algorithms using the degree of imbalance

Algorithm	100	200	300	400	500	600	700	800	900	1000	2000
GA	1.6250	1.3025	1.3252	1.7362	1.8520	1.8525	2.1500	2.3322	2.4411	2.4416	2.9251
DSOS	1.7500	1.3511	1.3526	1.6200	1.7522	1.7141	1.7545	1.8025	1.8055	1.8654	2.2565
MSDE	0.8649	0.9046	0.9321	0.8469	1.0228	1.0017	0.9548	0.9325	0.9895	1.0010	1.9541
PSO	1.2773	1.1988	0.9916	0.9564	1.0591	1.0112	1.0014	1.3147	1.2296	0.9985	1.8953
WOA	1.0857	1.1122	1.0624	0.9105	0.9576	1.0312	1.1451	0.9658	0.9965	1.0865	1.1450
MSA	1.2391	1.1039	0.9807	0.8432	0.9633	0.9010	0.9585	1.2440	1.1900	0.9874	1.0324
ALO	1.3253	1.2590	1.1995	1.0002	1.3541	1.4560	1.6540	1.1385	1.1221	0.9996	1.4251
MALO	0.9301	0.9561	0.9798	0.9020	0.9529	0.9263	0.9667	0.9216	0.9726	0.9701	1.3671

Table 7 Comparison of makespan obtained by basic ALO and MALO generated using different size of tasks

Task size	ALO			MALO			Improvement			*t*-test	
	Best	Worst	Avg	Best	Worst	Avg	Best (%)	Worst (%)	Avg (%)	*t*-value	*p*-value
100	72	79	76	64	75	70	− 5.56	+ 5.06	+ 7.89	+ 5.5796	0.0001
200	112	127	121	109	126	120	+ 2.68	+ 0.79	+ 0.83	+ 2.1693	0.0211
300	254	276	262	229	252	240	+ 9.84	+ 8.70	+ 8.40	+ 12.0503	0.0001
400	341	358	351	318	335	323	+ 6.74	+ 6.42	+ 7.98	+ 13.1664	0.0001
500	435	447	441	436	458	446	− 0.23	− 2.46	− 1.13	− 3.0690	0.0033
600	542	559	553	527	557	543	+ 2.77	+ 0.36	+ 1.81	+ 10.2841	0.0001
700	615	639	630	609	632	620	+ 0.98	+ 1.10	+ 1.59	+ 1.3063	0.1049
800	724	748	740	703	731	720	+ 2.90	+ 2.27	+ 2.70	+ 10.2533	0.0001
900	827	846	840	796	836	810	+ 3.75	+ 1.18	+ 3.57	+ 8.1241	0.0011
1000	904	929	921	894	925	900	+ 1.11	+ 0.43	+ 2.28	+ 6.4584	0.0001
2000	1750	1953	1842	1680	1845	1757	+ 1.04	+ 1.05	+ 1.04	+ 3.4504	0.0145

posed MALO and original ALO for these task instances. But, the other cases (500 tasks) have no significant improvement. Thus, the main goal here is to find a small makespan value and maximum resources utilization.

Moreover, the comparison results of degree of imbalance among the proposed MALO algorithm and the other comparative algorithms (GA, DSOS, MSDE, PSO, WOA, MSA, and ALO) are given in Fig. 5. It can be observed from these figures that the proposed MALO algorithm achieved better system load balance in comparison with the others. MALO got the smallest degree of imbalance, while, the other comparative optimization algorithms are competitive together. Also, it gave a better degree of imbalance among virtual machines for all problem instances as can be observed.

Figure 6 showed the average makespan for executing the small task instances (100-1000 tasks) using the tasks scheduling optimization algorithms (i.e., GA, DSOS, MSDE, PSO, WOA, MSA, ALO, and the proposed algorithm (MALO)). The average makespan for executing the large task instances (1000-2000 tasks) using the tasks scheduling optimization algorithms (i.e., GA, DSOS, MSDE, PSO, WOA, MSA, ALO, and the proposed algorithm (MALO)) are shown in Fig. 7. These two figures indicated that the proposed tasks scheduling algorithm (MALO) got better results in terms of the makespan measure, which means that MALO got the minimization of makespan values in most cases either small or large as can be seen in Figs. 6 and 7.

The comparison results between the original ALO and the proposed MALO algorithm according to the fitness function values are given in Figs. 8, 9 and 10. From these figures (results), it can be seen that the proposed algorithm (MALO) got better results than the original ALO in terms of performance measure (Fitness function). For instance, based on the average values of the fitness function, the proposed MALO algorithm reached the smallest values in all task cases such as shown in these Figs. 8 9 and 10. Moreover, the stability of the proposed algorithm (MALO) is clearly observed during solving all sizes of the tasks scheduling problem, which proved that using the proposed multi-objective function as well as the proposed hybrid

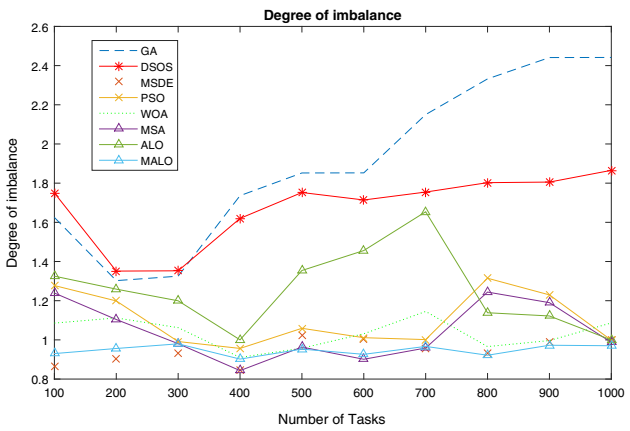


Fig. 5 The degree of imbalance of scheduling algorithms

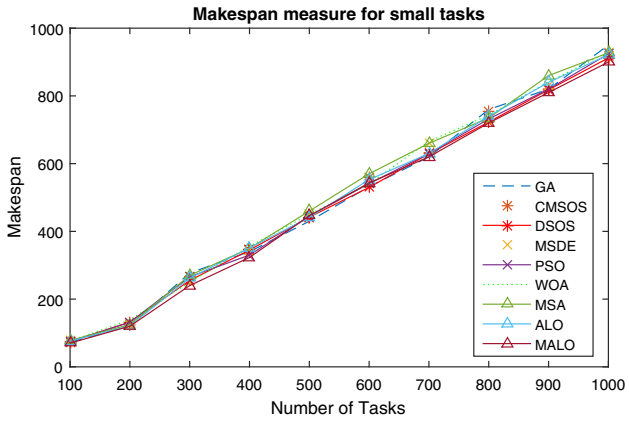


Fig. 6 The average makespan values for executing small tasks

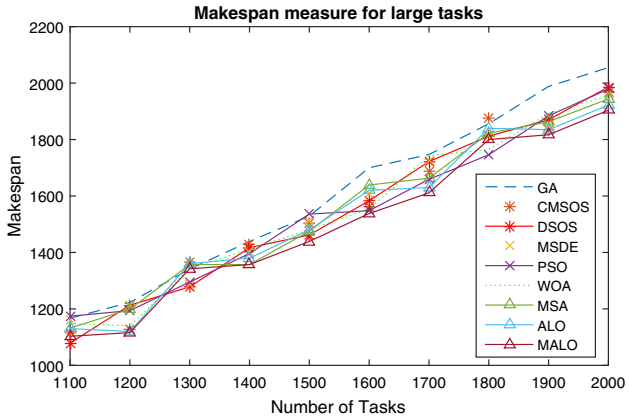


Fig. 7 The average makespan values for executing large tasks

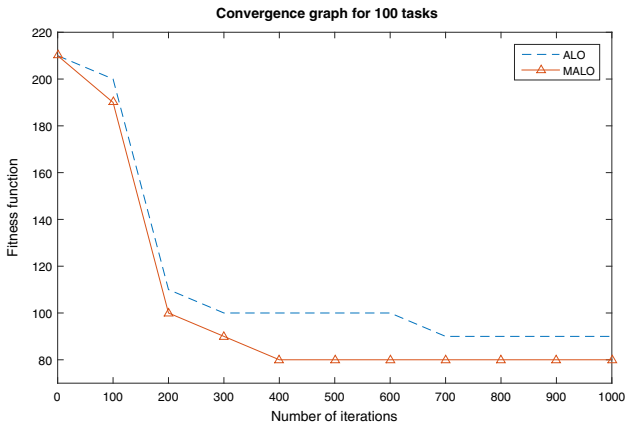


Fig. 8 The convergence curves for the fitness functions of MALO and ALO for 100 tasks

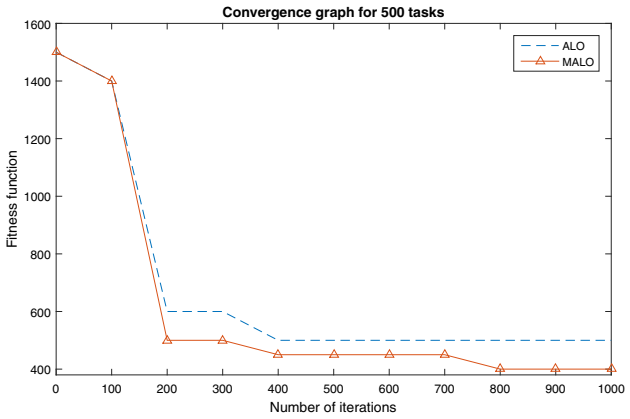


Fig. 9 The convergence curves for the fitness functions of MALO and ALO for 500 tasks

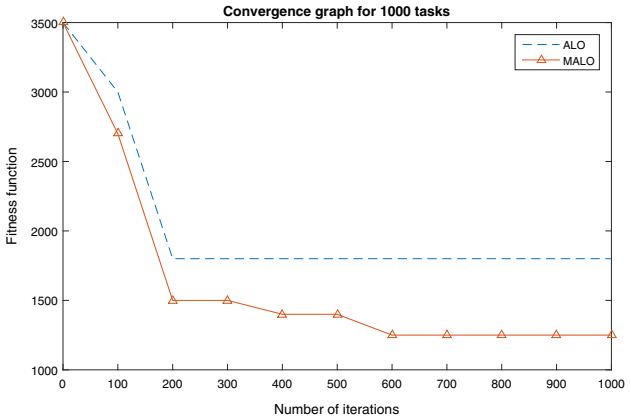


Fig. 10 The convergence curves for the fitness functions of MALO and ALO for 1000 tasks

version using deferential evolution accomplish an efficient method to solve the problem of the tasks scheduling.

Figure 11 showed the response times (CPU) obtained by the task scheduling algorithms (i.e., GA, DSOS, MSDE, PSO, WOA, MSA, ALO, and the proposed MALO) to perform its task. The figure displayed that the proposed MALO reached minimal response time for solving various sizes of the tasks scheduling problem in comparison with all other methods. The enhancement in the proposed algorithm (MALO) helps to reduce the required time to find the optimal solution. However, in case of the task size of 600, PSO algorithm got the minimal response time for solving this size of the tasks scheduling problem in comparison with all other methods.

5.3 Experiments part 2: evaluation results of real trace datasets

The most reliable approach to assess task scheduling approaches is to use realistic cloudlets (task/job) mixes,

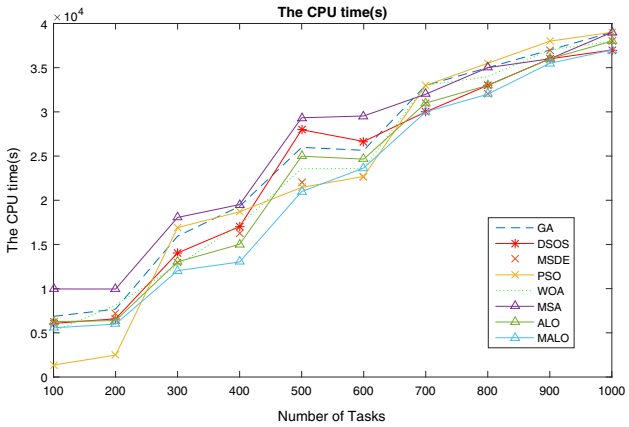


Fig. 11 The CPU time(s) of the task scheduling algorithms for the synthetic datasets

obtained from large-scale supercomputing situations. In this part of experiments, the introduced optimization algorithm (MALO) and the other comparative methods are evaluated by simulations utilizing the CloudSim to produce a cloud structure that includes corresponding diverse virtual machines with many of CloudSim's tasks. The details of tasks and virtual machines are obtained from a real-world log-traces (HPC2N Seth log-trace [82]) to model high-performance computing jobs/tasks and Feitelson's Parallel Workloads Archive (NASA Ames iPSC/860 log [83]). More details are given in Table 7.

The improvement of the proposed MALO compared with other comparative well-knowing optimization methods using two real trace datasets (i.e., HPC2N Seth dataset and NASA Ames dataset) is presented in Tables 8 and 9. In Table 8, it is observed clearly that the obtained results of the proposed MALO are better in comparison with all other comparative methods using HPC2N Seth dataset. In Table 9, the obtained results of the proposed MALO using NASA Ames dataset are better than all other comparative methods.

Figures 12 and 13 showed the response times (CPU) obtained by the task scheduling algorithms (i.e., GA, DSOS, MSDE, PSO, WOA, MSA, ALO, and the proposed MALO) to perform its task (i.e., real trace datasets). In Fig. 12, the proposed MALO algorithm almost reached the minimal response time in solving all sizes of tasks in comparison with the other comparative methods using the HPC2N Seth datasets. As well, in Fig. 13, the proposed MALO algorithm almost reached the minimal response time in solving all sizes of tasks in comparison with the other comparative methods using the NASA Ames datasets. Moreover, the difference between the response times of the proposed algorithm over all the tasks sizes is clear compared with other methods using the HPC2N Seth datasets. But, the difference between the response times of

Table 8 Comparisons of the improvement ratio between the MALO and other comparative methods using HPC2N Seth datasets

Algorithm	100 (%)	200 (%)	300 (%)	400 (%)	500 (%)	600 (%)	1000 (%)	2000 (%)
GA	6.32	4.32	4.65	6.27	10.65	5.01	21.58	11.5
DSOS	7.36	5.25	4.69	7.25	10.11	4.68	17.21	15.10
PSO	12.96	3.73	8.43	7.48	10.29	6.27	1.41	2.40
WOA	7.78	7.40	5.52	6.14	9.09	8.55	2.20	2.14
MSA	11.30	2.59	3.44	8.23	9.41	5.04	10.17	6.98
ALO	5.12	5.69	4.69	6.95	9.11	5.17	4.58	6.25
MALO	12.25	7.85	5.60	8.11	8.54	9.25	12.95	16.55

Table 9 Comparisons of the improvement ratio between the MALO and other comparative methods using NASA Ames datasets

Algorithm	100 (%)	200 (%)	300 (%)	400 (%)	500 (%)	600 (%)	1000 (%)	2000 (%)
GA	1.20	5.25	4.39	2.58	1.67	3.21	22.52	10.12
DSOS	1.25	20.66	19.25	5.25	10.69	10.45	16.21	16.22
PSO	3.59	7.86	10.19	5.06	2.99	7.56	0.64	7.65
WOA	2.67	10.66	1.58	3.16	6.06	9.42	0.75	2.65
MSA	1.13	22.20	25.51	6.42	11.77	20.72	7.45	8.52
ALO	1.25	5.25	3.55	2.34	2.69	3.58	2.97	3.01
MALO	4.25	8.65	11.69	7.14	12.90	11.20	17.56	17.01

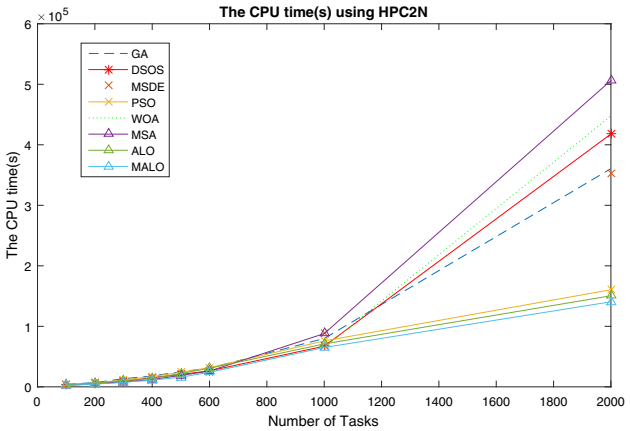


Fig. 12 The CPU time(s) of the task scheduling algorithms for solving the HPC2N Seth datasets

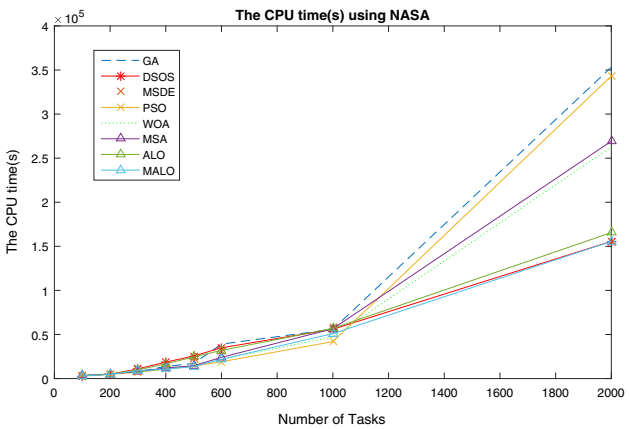


Fig. 13 The CPU time(s) of the task scheduling algorithms for solving the NASA Ames datasets

the proposed algorithm over all the tasks sizes is not clear (it is similar to the response time of the DSOS) compared with other methods using the HPC2N Seth datasets. Finally, the enhancement in the proposed algorithm (MALO) helps to reduce the required time in finding the optimal solution.

The comparison results of degree of imbalance among the proposed MALO algorithm in comparison with the other comparative algorithms (GA, DSOS, MSDE, PSO, WOA, MSA, and ALO) are given in Fig. 14 for the HPC2N Seth datasets, and the degrees of imbalance are given in Fig. 15. It can be observed from these figures that the proposed MALO algorithm achieved better system load balance (degree of imbalance) in comparison with the other methods. MALO got the smallest degree of imbalance almost in all the datasets cases (100–2000), while the other comparative optimization algorithms are competitive together. Also, it gave a better degree of imbalance among virtual machines for all problem instances as can be observed.

6 Conclusion and future works

We presented a novel hybrid antlion optimization algorithm with elite-based differential evolution for solving multi-objective task scheduling problems in cloud computing environments. Two experimental series were conducted on synthetic and real trace datasets using the CloudSim tool kit. Response time (CPU), degree of imbalance, and makespan were measured for each algorithm. The results revealed that MALO outperformed other

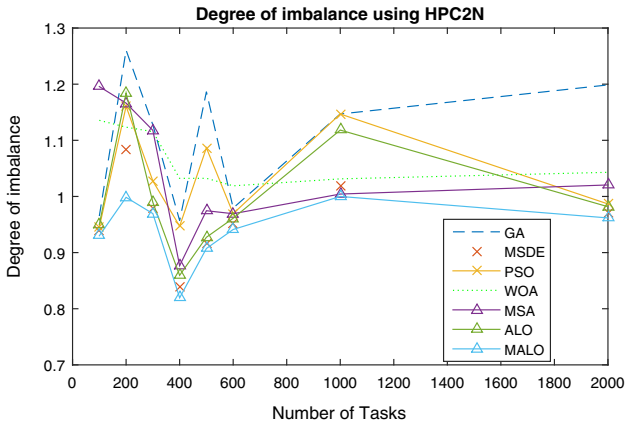


Fig. 14 The degree of imbalance of the tasks scheduling optimization algorithms using the HPC2N Seth datasets

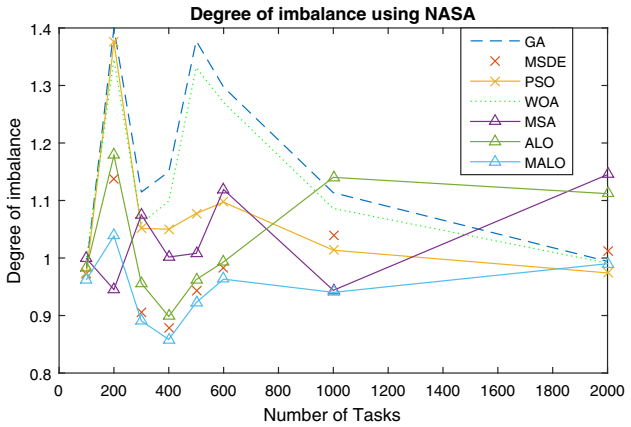


Fig. 15 The degree of imbalance of the tasks scheduling optimization algorithms using the NASA Ames datasets

well-known algorithms for solving the task scheduling problem. MALO converged faster than the other approaches for larger search spaces, making it suitable for large scheduling problems. Finally, the results were analyzed using statistical t-tests, which showed that MALO obtained a significant improvement in the results.

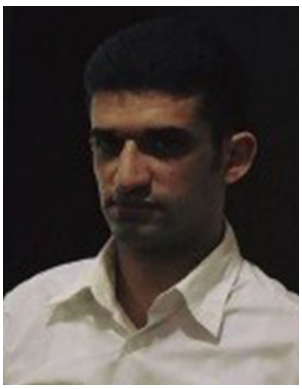
In the future, we plan to compare the new method with other existing methods to validate its performance, and to eventually improve its time complexity. Application of MALO to other optimization problems is also a possible future research direction. Furthermore, MALO can be extended to consider other parameters in the cloud computing environment such as the peak demand, memory use, and overloads.

References

1. Kumar, M., Sharma, S., Goel, A., Singh, S.: A comprehensive survey for scheduling techniques in cloud computing. J. Netw. Comput. Appl. (2019). https://doi.org/10.1016/j.jnca.2019.06.006
2. Natesan, G., Chokkalingam, A.: An improved grey wolf optimization algorithm based task scheduling in cloud computing environment. Int. Arab J. Inf. Technol. **17**(1), 73–81 (2017)
3. Abdullahi, M., Ngadi, M.A., Dishing, S.I., Ahmad, B.I., et al.: An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment. J. Netw. Comput. Appl. **133**, 60–74 (2019)
4. Mohammadi, A., Rezvani, M.H.: A novel optimized approach for resource reservation in cloud computing using producer-consumer theory of microeconomics. J. Supercomput. **75**, 7391–7425 (2019)
5. Geng, X., Yu, L., Bao, J., Fu, G.: A task scheduling algorithm based on priority list and task duplication in cloud computing environment. Web Intell. **17**, 121–129 (2019)
6. Beegom, A.A., Rajasree, M.: Integer-pso: a discrete pso algorithm for task scheduling in cloud computing systems. Evol. Intell. **12**, 227–239 (2019)
7. Abualigah, L.M., Khader, A.T., Hanandeh, E.S.: A new feature selection method to improve the document clustering using particle swarm optimization algorithm. J. Comput. Sci. **25**, 456–466 (2018a)
8. Abualigah, L.M., Khader, A.T., Hanandeh, E.S.: Hybrid clustering analysis using improved krill herd algorithm. Appl. Intell. **48**, 4047–4071 (2018b)
9. Shehab, M., Daoud, M.S., AlMimi, H.M., Abualigah, L.M., Khader, A.T.: Hybridising cuckoo search algorithm for extracting the odf maxima in spherical harmonic representation. Int. J. Bio-Inspir. Comput. **14**, 190–199 (2019)
10. Rodrigues, L.R., Gomes, J.P.P.: Tlbo with variable weights applied to shop scheduling problems. CAAI Trans. Intell. Technol. **4**, 148–158 (2019)
11. Gaurav, D., Tiwari, S.M., Goyal, A., Gandhi, N., Abraham, A.: Machine intelligence-based algorithms for spam filtering on document labeling. Soft Comput. (2019). https://doi.org/10.1007/s00500-019-04473-7
12. Mishra, S., Sagban, R., Yakoob, A., Gandhi, N.: Swarm intelligence in anomaly detection systems: an overview. J. Comput. Appl, Int (2018). https://doi.org/10.1080/1206212X.2018.1521895
13. Abualigah, L.M., Khader, A.T.: Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering. J. Supercomput. **73**, 4773–4795 (2017)
14. Abualigah, L.M., Khader, A.T., Hanandeh, E.S.: A combination of objective functions and hybrid krill herd algorithm for text document clustering analysis. Eng. Appl. Artif. Intell. **73**, 111–125 (2018)
15. Abualigah, L.M., Khader, A.T., Hanandeh, E.S., Gandomi, A.H.: A novel hybridization strategy for krill herd algorithm applied to clustering techniques. Appl. Soft Comput. **60**, 423–435 (2017)
16. Shehab, M., Abualigah, L., AlHamad, H., Alabool, H., Alshinwan, M., Khasawneh, A.M.: Moth-flame optimization algorithm: variants and applications. Neural Comput. Appl. (2019). https://doi.org/10.1007/s00521-019-04570-6
17. Abualigah, L.M.Q., Hanandeh, E.S.: Applying genetic algorithms to information retrieval using vector space model. Int. J. Comput. Sci. Eng. Appl. **5**, 19 (2015)
18. Zheng, Y.-J., Xu, X.-L., Ling, H.-F., Chen, S.-Y.: A hybrid fireworks optimization method with differential evolution operators. Neurocomputing **148**, 75–82 (2015)
19. Yazdi, J., Choi, Y.H., Kim, J.H.: Non-dominated sorting harmony search differential evolution (ns-hs-de): a hybrid algorithm for multi-objective design of water distribution networks. Water **9**, 587 (2017)

20. Li, Y., Li, X., Li, Z.: Reactive power optimization using hybrid cabc-de algorithm. Electr. Power Compon. Syst. **45**, 980–989 (2017)
21. Zhang, L., Liu, L., Yang, X.-S., Dai, Y.: A novel hybrid firefly algorithm for global optimization. PLoS ONE **11**, e0163230 (2016)
22. Kumar, K.P., Kousalya, K.: Amelioration of task scheduling in cloud computing using crow search algorithm. Neural Comput. Appl. (2019). https://doi.org/10.1007/s00521-019-04067-2
23. Matos, J.G.D., Marques, C.K.D.M., Liberalino, C.H.: Genetic and static algorithm for task scheduling in cloud computing. Int. J. Cloud Comput. **8**, 1–19 (2019)
24. Thanka, M.R., Maheswari, P.U., Edwin, E.B.: A hybrid algorithm for efficient task scheduling in cloud computing environment. Int. J. Reason. Based Intell. Syst. **11**, 134–140 (2019)
25. Abualigah, L.M.Q.: Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering. Springer, Berlin (2019)
26. Domingo, M., Thibaud, R., Claramunt, C.: A graph-based approach for the structural analysis of road and building layouts. Geo-spatial Inf. Sci. **22**, 59–72 (2019)
27. Kashikolaei, S.M.G., Hosseinabadi, A.A.R., Saemi, B., Shareh, M.B., Sangaiah, A.K., Bian, G.-B.: An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm. J. Supercomput **1**, 28 (2019). https://doi.org/10.1007/s11227-019-02816-7
28. Mapetu, J.P.B., Chen, Z., Kong, L.: Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing. Appl. Intell. **49**, 3308–3330 (2019)
29. Zhou, Z., Chang, J., Hu, Z., Yu, J., Li, F.: A modified pso algorithm for task scheduling optimization in cloud computing. Concurr. Comput. **30**, e4970 (2018)
30. Yassa, S., Chelouah, R., Kadima, H., Granado, B.: Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. Sci. World J. (2013). https://doi.org/10.1155/2013/350934
31. Alla, H.B., Alla, S.B., Ezzati, A., Mouhsen, A.: A novel architecture with dynamic queues based on fuzzy logic and particle swarm optimization algorithm for task scheduling in cloud computing. In: International Symposium on Ubiquitous Networking, Springer, pp. 205–217 (2016)
32. Agarwal, M., Srivastava, G.M.S.: A PSO algorithm-based task scheduling in cloud computing. In: Soft Computing: Theories and Applications. Springer, pp. 295–301 (2019)
33. Abdullahi, M., Ngadi, M.A., et al.: Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Gener. Comput. Syst. **56**, 640–650 (2016)
34. Elaziz, M.A., Xiong, S., Jayasena, K., Li, L.: Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. Knowl.-Based Syst. **169**, 39–52 (2019)
35. Zuo, L., Shu, L., Dong, S., Zhu, C., Hara, T.: A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. IEEE Access **3**, 2687–2699 (2015)
36. Moon, Y., Yu, H., Gil, J.-M., Lim, J.: A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments. Hum.-Centric Comput. Inf. Sci. **7**, 28 (2017)
37. Agarwal, M., Srivastava, G.M.S.: Genetic algorithm-enabled particle swarm optimization (PSOGA)-based task scheduling in cloud computing environment. Int. J. Inf. Technol. Decis. Mak. **17**, 1237–1267 (2018)
38. Nzanywayingoma, F., Yang, Y.: Analysis of particle swarm optimization and genetic algorithm based on task scheduling in cloud computing environment. Int. J. Adv. Comput. Sci. Appl. **8**, 19–25 (2017)
39. Zheng, X.-L., Wang, L.: A pareto based fruit fly optimization algorithm for task scheduling and resource allocation in cloud computing environment. In: IEEE Congress on Evolutionary Computation (CEC). IEEE 2016, pp. 3393–3400 (2016)
40. Mansouri, N., Javidi, M.: Cost-based job scheduling strategy in cloud computing environments. Distrib. Parallel Databases (2016). https://doi.org/10.1007/s10619-019-07273-y
41. Abdullahi, M., Dishing, S.I., Usman, M.J., et al.: Variable neighborhood search-based symbiotic organisms search algorithm for energy-efficient scheduling of virtual machine in cloud data center. In: Advances on Computational Intelligence in Energy, Springer, pp. 77–97 (2019)
42. Taherian Dehkordi, S., Khatibi Bardsiri, V.: Optimization task scheduling algorithm in cloud computing. J. Adv. Comput. Eng. Technol. **1**, 17–22 (2015)
43. Saxena, D., Chauhan, R., Kait, R.: Dynamic fair priority optimization task scheduling algorithm in cloud computing: concepts and implementations. Int. J. Comput. Netw. Inf. Secur. **8**, 41 (2016)
44. Rani, E., Kaur, H.: Efficient load balancing task scheduling in cloud computing using raven roosting optimization algorithm. Int. J. Adv. Res. Comput. Sci **8**, 2419–2424 (2017)
45. Alazzam, H., Alhenawi, E., Al-Sayyed, R.: A hybrid job scheduling algorithm based on tabu and harmony search algorithms. J. Supercomput. **75**, 7994–8011 (2019)
46. Valarmathi, R., Sheela, T.: Ranging and tuning based particle swarm optimization with bat algorithm for task scheduling in cloud computing. Clust. Comput. **22**, 11975–11988 (2017)
47. Gawali, M.B., Shinde, S.K.: Standard deviation based modified cuckoo optimization algorithm for task scheduling to efficient resource allocation in cloud computing. J. Adv. Inf. Technol (2017). https://doi.org/10.12720/jait.8.4.210-218
48. Sundarrajan, R., Vasudevan, V.: An optimization algorithm for task scheduling in cloud computing based on multi-purpose cuckoo seek algorithm. In: International Conference on Theoretical Computer Science and Discrete Mathematics, Springer, pp. 415–424 (2016)
49. Dai, Y., Lou, Y., Lu, X.: A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-qos constraints in cloud computing. In: 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics, vol. 2, pp. 428–431. IEEE (2015)
50. Abubakar, A., Yahaya, A.: Task scheduling in cloud computing environment using particle swarm optimization algorithm. Niger. J. Sci. Res. **14**, 106 (2015)
51. Liu, Y., Shu, W., Zhang, C.: A parallel task scheduling optimization algorithm based on clonal operator in green cloud computing. J. Commun. **11**, 185–191 (2016)
52. Varshney, S., Singh, S.: An optimal bi-objective particle swarm optimization algorithm for task scheduling in cloud computing. In: 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), IEEE, pp. 780–784 (2018)
53. An, J.H., Lim, C.H., Cho, Y.C., Lee, C.S.: Early recovery process and restoration planning of burned pine forests in central eastern korea. J. For. Res. **30**, 243–255 (2019)
54. Saranu, K., Jaganathan, S.: Intensified scheduling algorithm for virtual machine tasks in cloud computing. In: Suresh, L., Dash, S., Panigrahi, B. (eds.) Artificial Intelligence and Evolutionary Algorithms in Engineering Systems, pp. 283–290. Springer, Berlin (2015)
55. Al-Rahayfeh, A., Atiewi, S., Abuhussein, A., Almiani, M.: Novel approach to task scheduling and load balancing using the dominant sequence clustering and mean shift clustering algorithms. Future Internet **11**, 109 (2019)

56. 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)
57. Li, Y., Wang, S., Hong, X., Li, Y.: Multi-objective task scheduling optimization in cloud computing based on genetic algorithm and differential evolution algorithm. In: 2018 37th Chinese Control Conference (CCC), IEEE, pp. 4489–4494 (2018)
58. Masadeh, R., Sharieh, A., Mahafzah, B.: Humpback whale optimization algorithm based on vocal behavior for task scheduling in cloud computing. Int. J. Adv. Sci. Technol. **13**, 121–140 (2019)
59. Luo, F., Yuan, Y., Ding, W., Lu, H.: An improved particle swarm optimization algorithm based on adaptive weight for task scheduling in cloud computing, in: Proceedings of the 2nd International Conference on Computer Science and Application Engineering, ACM, p. 142 (2018)
60. Reddy, G.N., Kumar, S.P.: Modified ant colony optimization algorithm for task scheduling in cloud computing systems. In: Satapathy, S., Bhateja, V., Das, S. (eds.) Smart Intelligent Computing and Applications, pp. 357–365. Springer, Berlin (2019)
61. Demiroz, B., Topcuoglu, H.R.: Static task scheduling with a unified objective on time and resource domains. Comput. J. **49**, 731–743 (2006)
62. Loo, S.M., Wells, B.E.: Task scheduling in a finite-resource, reconfigurable hardware/software codesign environment. INFORMS J. Comput. **18**, 151–172 (2006)
63. Rahul, M.: An efficient multi-objective genetic algorithm for optimization of task scheduling in cloud computing. Asian J. Technol. Manag. Res. [ISSN: 2249–0892] (2016)
64. Zhang, F., Cao, J., Li, K., Khan, S.U., Hwang, K.: Multi-objective scheduling of many tasks in cloud platforms. Future Gener. Comput. Syst. **37**, 309–320 (2014)
65. Mirjalili, S.: The ant lion optimizer. Adv. Eng. Softw. **83**, 80–98 (2015)
66. Cuevas, E., Echavarría, A., Ramírez-Ortegón, M.A.: An optimization algorithm inspired by the states of matter that improves the balance between exploration and exploitation. Appl. Intell. **40**, 256–272 (2014)
67. Yang, X.-S.: A new metaheuristic bat-inspired algorithm. In: Gonzalez, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) Nature Inspired Cooperative Strategies for Optimization, pp. 65–74. Springer, Berlin (2010)
68. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**, 182–197 (2002)
69. Yang, X.-S.: Flower pollination algorithm for global optimization. In: International conference on unconventional computing and natural computation, Springer, pp. 240–249 (2012)
70. Yang, X.-S., Deb, S.: Cuckoo search via lévy flights. In: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, pp. 210–214 (2009)
71. Kennedy, J.: Particle swarm optimization. Encyclopedia of machine learning, pp. 760–766 (2010)
72. Yang, X.-S.: Firefly algorithm, levy flights and global optimization. In: Bramer, M., Ellis, R., Petridis, M. (eds.) Research and Development in Intelligent Systems, vol. 26, pp. 209–218. Springer, Berlin (2010)
73. Hatata, A.Y., Hafez, A.A.: Ant lion optimizer versus particle swarm and artificial immune system for economical and eco-friendly power system operation. Int. Trans. Electr. Energy Syst. **29**, e2803 (2019)
74. Roy, K., Mandal, K.K., Mandal, A.C.: Ant-lion optimizer algorithm and recurrent neural network for energy management of micro grid connected system. Energy **167**, 402–416 (2019)
75. Wang, M., Wu, C., Wang, L., Xiang, D., Huang, X.: A feature selection approach for hyperspectral image based on modified ant lion optimizer. Knowl.-Based Syst. **168**, 39–48 (2019)
76. Storn, R., Price, K.: Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**, 341–359 (1997)
77. Humane, P., Varshapriya, J.: Simulation of cloud infrastructure using cloudsim simulator: A practical approach for researchers. In: 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), IEEE, pp. 207–211 (2015)
78. Zhang, L., Li, K., Li, C., Li, K.: Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems. Inf. Sci. **379**, 241–256 (2017)
79. Zuo, X., Zhang, G., Tan, W.: Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud. IEEE Trans. Autom. Sci. Eng. **11**, 564–573 (2013)
80. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016)
81. Wang, G.-G.: Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Comput. **10**, 151–164 (2018)
82. Feitelson, D.G., Tsafrir, D., Krakov, D.: Experience with using the parallel workloads archive. J. Parallel Distrib. Comput. **74**, 2967–2982 (2014)
83. Meng, J., McCauley, S., Kaplan, F., Leung, V.J., Coskun, A.K.: Simulation and optimization of HPC job allocation for jointly reducing communication and cooling costs. Sustain. Comput. **6**, 48–57 (2015)



Laith Abualigah is an Assistant Professor at the Software Engineering Department, Amman Arab University, Jordan. He received his first degree from Al-Albayt University, Computer Information System, Jordan, in 2011. He earned a Master's degree from Al-Albayt University, Computer Science, Jordan, in 2014. He received the Ph.D. degree from the School of Computer Science in Universiti Sains Malaysia (USM), Malaysia in 2018. His main research interests focus on Bio-inspired Computing, Artificial Intelligence, Metaheuristic Modeling, and Optimization Algorithms, Evolutionary Computations, Information Retrieval, Feature Selection, Combinatorial Problems, Optimization, Advanced Machine Learning, Big data, and Natural Language Processing.

Ali Diabat is a Global Network Professor of Engineering at New York University Abu Dhabi. His research focuses on different applications of optimization and operations research: in particular, logistics and supply chain management, healthcare management, and production planning. He has published over 100 research journal papers and over 30 conference papers in leading journals and international conference proceedings, and his research has received about 5 million dollars of grant funding from different industries and collaborative proposals with academic institutions. Dr. Diabat has received several Excellence in Teaching awards, including an Outstanding Graduate Instructor Award and an Excellence in Teaching Award from Purdue University, in 2004 and 2006, respectively. He was the recipient of the 2012 Excellence in Teaching Award from Masdar Institute, and in 2014, he also received the Best Faculty Research Award from the Department of Engineering Systems and Management at Masdar Institute. Dr. Diabat currently serves as an Associate Editor of the Journal of Manufacturing Systems, an Area Editor of the Journal of Computers and Industrial Engineering, and an Engineering Editor of the Arabian Journal for Science and Engineering.