Received: September 23, 2021. Revised: November 9, 2021.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Automated Task Scheduling Model Using a Multi-objective Improved Cuckoo Optimization Algorithm

Sajjad Jaber[1]* **Yossra Ali[1]** **Nuha Ibrahim[1]**

[1]*Department of Computer Science, University of Technology, Iraq*
* Corresponding author's Email: sajjad.sh.jaber@gmail.com

Abstract: Cloud computing is a new computing paradigm that consists of a large number of heterogeneous autonomous systems that have a flexible computational architecture, task scheduling is extremely significant, this process must schedule jobs on a virtual machine while minimizing time and cost. The task scheduling problem is classified as NP-hard. The use of an efficient scheduling approach improves and speeds up cloud computing services. In general, optimization strategies are employed to overcome cloud scheduling issues. In this paper, we first propose an optimization model based on a Multi-Objective Improved Cuckoo Search Algorithm (MOICS) to optimize task scheduling problems in a cloud environment this reduces both the time it takes to process the tasks and the overall cost. Then there's the discrete multi-objective task scheduling problem to solve, as well as automatically assigning work to cloud nodes. The suggested methodology allocates computational resources that can be used effectively on cloud nodes. After the implementation of the proposed method, the results show that our proposed work minimize the makespan and cost when compared to Modified Particle Swarm Optimization (MPSO), Bee Life Algorithm (BLA), A Time–Cost aware Scheduling (TCaS) algorithm, and Round Robin (RR) algorithm.

Keywords: Task scheduling, Multi-objective optimization, Cuckoo search, Cloud computing.

1. Introduction

When we consider the vast amount of data created by IoT devices, this becomes even more important. In smart city applications, as an example, these linked devices generate vast volumes of data, which must be collected, handled, and analyzed to extract meaningful insights, in addition, to correctly accessed by end-users and/or client apps. Alongside, the number and services and apps grow, needing computational powers that exceed the capability of even the absolute most powerful smart devices. Meantime, cloud computing, which offers dynamically scaled and often iterative services as an online service, can overcome these IoT-related issues [1]. Task scheduling is an essential problem in the cloud computing environment since it takes into account a variety of aspects such as completion time, the total cost of executing all users' activities, power consumption, resource utilization, and fault tolerance. Scheduling is a way of making decisions and plays an important role in most manufacturing and production systems as well as most of the information processing environments that are used daily in a variety of industrial settings [2, 3]. The task scheduler assigns compute resources to cloud nodes, while the load balancer distributes workloads over different computing resources [4]. Depending on their interrelationship between tasks may appear in different models. One kind of tasks can be processed in parallel on a single processing level by using bags of tasks (BoT), where the tasks do not depend on one another in any way. Conversely, workflow tasks can be interconnected by narrow pathways, so that the execution of one subtask can be dependent on the execution of one or more subtasks at a previous level [5].

Since the cloud scheduling issue is non-deterministic Polynomial-time hard (NP-hard) [6], which cannot be efficiently solved by classic methods. As a result, recent studies have increased the research into heuristic and metaheuristic algorithms in order to optimize task scheduling due to their effectiveness in solving problems of high complexity and large size [7]. In several studies, the BoT scheduling problem has been formulated as a single objective problem to solve the problem of cost [8] and makespan [9] optimization under certain constraints, such as execution deadlines and task budgets. In addition, due to the complex nature of workflow tasks, several multiobjective metaheuristic methods have been developed to find near optimal scheduling solutions by considering conflicted optimization goals, namely, costs and makespan [10].

The focus of this research is on task scheduling challenges in the cloud, a highly distributed computing architecture for handling huge BoT applications. A multi-objective Improved Cuckoo Search (MOICS) algorithm is developed to address this problem. The main purpose of the MOICS algorithm is to achieve a fair balance between execution time and financial cost when completing a set of activities in the cloud system . Furthermore, this algorithm is flexible enough to adjust to the requirements of different users to meet their goals. Some users prioritize execution times, while others prioritize budgets. On a variety of datasets of different sizes, our method was tested and compared to the MPSO algorithm, BLA Algorithm, TCaS Algorithm, and RR algorithm. The results demonstrate that the proposed algorithm provided the best Quality of Service and was faster and cheaper than the other strategies. The contributions for the MOICS algorithm are:

- Our approach formulates the task-scheduling problem as a multi-objective optimization problem in a cloud system, intending to reduce total execution durations and costs by dynamic allocating appropriate resources to present tasks.
- For discrete multi-objective optimization issues, such as allocating specific compute nodes that can't be allocated utilizing a continuous space, we propose an enhanced Multi-Objective Cuckoo search optimization model.

The following is how the rest of the article is structured. The related works and background are presented in Section 2. Section 3 describes the formulation for the work scheduling problem in the Cloud computing context. Section 4 delves into the details of our proposed algorithm. The results of the experiments are presented in Section 5. Finally, Section 6 brings the article to a close and discusses potential work.

2. Related work and background

This part describes task scheduling and resource allocation principles and studies in a cloud setting, as well as multi-objective optimization and the Cuckoo search algorithm.

2.1 Task scheduling

Cloud computing resources must be allocated not just to meet user-specified QoS standards, but also to shorten the time to execute, lower costs, and increase service provider profitability. Scheduling and load balancing techniques are stringent for increasing Cloud computing efficiency while using limited resources [11, 12]. Dependent and independent approaches to task scheduling in computing platforms are widely categorized [13]. There are dependencies and communications when distributing and assign tasks to specific compute resources in the based methodologies. Individual tasks are distributed across computing exchequer in different ways, either in batch mode or in real-time [13]. Because it is more useful and closely matches the job arrival and allocation processes in the actual world, the online mode was chosen for our architecture.

Work scheduling algorithms are essential in complicated environments for allocating batch mode jobs and coordinating the access time of each task scheduled for the resource management system. The task scheduling has to be expressed as being an NP-hard issue [14, 15].

According to [16], a large group of Internet of Things units was published between 2016 to 2020, which led to the emergence of problems in cloud environments. Researchers tried to solve these problems through several types of research to assign tasks in cloud systems. [17] concentrated their efforts on reducing latency in edge Clouds. In Cloud–Fog computing, they proposed a work allocation problem. They did, however, calculate power consumption and latency using a simple model. In another study, [18] have been suggested the Bee Life Algorithm (BLA) as a task scheduling technique. The emphasis of the study is on key goals: memory and execution time. However, the connection with Cloud data centers is not mentioned in this article, and the approach has only been tested on small datasets. while in [19] a hybrid scheduling algorithm that mimics cuckoo's parasitic behaviour and the crow's food gathering behavior, named Cuckoo Crow Search Algorithm (CCSA), has been proposed to address cloud computing's task scheduling challenge. To optimize the performance of task scheduling, the hybrid CCSA incorporates several QoS parameters, including makespan and cost. however, the tests were restricted because the algorithm was only tested on small datasets. In another study, [1] when scheduling large-scale applications in such a platform, investigated the tradeoff between makespan and cloud cost and proposed the Cost-Makespan aware scheduling method. Despite the fact that their approach required a cost makespan barter, they linearized the tradeoff using a rational factor rather than Pareto optimization techniques. In [20] they formulated an associated task scheduling problem into a constraint to tackle the difficult task scheduling problem with some priority constraints of IoT applications while taking into account energy consumption and decreasing energy consumption under the condition of meeting the mixed deadline. To solve this problem, a laxity and ant colony system algorithm (LBP-ACS) is proposed. A task scheduling method in this algorithm takes into considers not only a work's priority but also its completed deadline. Further, they only tested their proposed approach on tiny datasets. Where [21] conducted a similar study and suggested the static task graph scheduling inhomogeneous multiprocessor environments, the predominant technology used as mini-servers in fog computing, is addressed using a high-performance approach based on the Max-Min Ant System (MMAS), which is an efficient variation in the family of ant colony optimization algorithms. In [22] Hybrid Max-Min Genetic Algorithms (HMMGA) have been proposed that can be used to handle load balancing and task scheduling issues in the cloud. HMMGA aims to reduce the completion time of heterogeneous VMs and complex scheduling decisions.In order to achieve effective task scheduling and load balancing, HMMGA defines two constraints, such as the earliest finish time and the optimal completion time. [23] presented a novel method for optimizing task scheduling in a Cloud–Fog environment for Bag-of-Tasks applications in terms of execution time and operational expenses.

As previously mentioned, most current task scheduling approaches are only relevant for small datasets and only consider cost or makespan as a single goal. Though some barter research has been done in the past, it is unlikely that those tradeoffs accurately reflect reality. Pareto optimization techniques, also known as multi optimization algorithms, are essential when dealing with two or more competing goals. To solve a multi-objective task scheduling problem, [24] suggested a hybrid genetic-ACO algorithm to solve a multi-objective task scheduling methodology.

The majority of today's task scheduling approaches are multiobjective issues, which are more workable than single-objective issues [25-27]. The multi-objective problem requires a tradeoff between several goals, to make the best decision possible. As a result, this study suggests a multi-objective task scheduling issue in cloud environments. To do so, we build an updated MOICS algorithm that is capable of solving multi-objective optimization problems efficiently.

2.2 Cuckoo search algorithm

Cuckoo Search (CS) is a fascinating bird, not just because of its wonderful sound, but also because of its aggressive reproduction technique. [28] suggested the CS metaheuristic algorithm. The algorithm's theoretical inspiration came from the vigorous reproduction technique of the cuckoo bird. Yang and Deb use three ideal rules to apply this technique as an optimization method [26, 29]:

- Each cuckoo lays one egg at a time and deposits it in a nest that is picked at random.
- A portion of the high-quality egg mite (best solutions) will be passed on to the next generation.
- The number of nests is set, and the host has a probability $P_a \in (0,1)$ of discovering an outsider egg, which results in the host eliminating the egg or nest and constructing a completely new nest in another location.

The key steps of the CS algorithm can be illustrated as the pseudocode shown in Algorithm 1 using these three laws. Using levy flight, a new solution $x^{(t+1)}$ for the ith cuckoo is produced as follows:

$$X_i^{(t+1)} = x_i^{(t)} + \alpha \oplus Levy(\beta) \quad (1)$$

The phase size that should be defined with the problem scales is $\alpha > 0$. The majority of the time will use $\alpha = 1$. The term product $\oplus$ refers to entry-by-entry multiplication. Levy flights essentially generate a random walk, with their random phase lengths drawn from the Levy distribution.

$$Levy \sim u = t^{-\beta} \quad (1 < \beta \leq 3) \quad (2)$$

which has an endless number of options The sequential steps/jumping off a cuckoo are simply a random walk with a strong tail that follows a power law step-length distribution. A fraction P_a of the poorest nests can be removed using random walk, and new nests can be created in new locations.

Since multi-objective optimization problems require simultaneous optimization of multiple

Received: September 23, 2021. Revised: November 9, 2021.

objective functions, and there is no additional information about the metaheuristic multiobjective issue to be solved, no single Pareto optimal solution can be viewed as superior to others [30]. As a result, the best solutions (decisions) must be decided as a barter among several goals, and the optimality of a solution varies depending on a variety of factors like the user preference, the issue description, and the context.

Algorithm 1. Cuckoo Search Algorithm

```
1: begin
2:   Objective function $f(x)$, $x=(x_1,\dots,x_n)^T$;
3:   Initial a population of n host nests $x_i(i=1,2,\dots,n)$;
4:     while (t < Maximum Generation) or (stop criterion);
5:       Get a cuckoo (say i) randomly and generate a new solution by Levy flights;
6:       Evaluate its quality/fitness $F_i$
7:       Choose a nest among n (say j) randomly;
8:       if ($F_i > F_j$)
9:           Replace j with a new solution;
10:      end
11:      Abandon a fraction ($P_a$) of worst nests;
12:      Keep the best solutions (or nests with quality solutions);
13:      Rank the solutions and find the current best;
14:    end while

15:   Post process results and visualization;
16: end
```

3. Formulation of task scheduling problem in cloud system

Table 1. The symbols used in the work

Symbol	Description
P_i	number machine i
T_j	number task j
M	the total amount of virtual machines available
N	the overall number of tasks
T_i^j	task j is processed by machine i
$I(T_j)$	total number of j task instructions
$RB(T_j)$	the required bandwidth for task j
$RM(T_j)$	required memory for task j
$S_r(P_i)$	computing average of machine i
$BW_c(p_i)$	cost of bandwidth usage for machine i
$S_c(p_i)$	computing cost for machine i
$M_c(p_i)$	cost of memory usage of machine i

When BoT apps make requests to the cloud layer, they're broken down into small, autonomous tasks which can be treated by the cloud computing infrastructure. Every task has its collection of characteristics, like the size of the I/O files, the numeral of instructions, and the amount of memory required.

Table 1 lists the notations for the majority of mathematical symbols. A typical task scheduling issue would aim to schedule all tasks to ensure that total execution times are as short as you can when using the fewest total costs of available resources. Let N be the number of tasks (T = T1, T2,..., Tj,..., TN) to be handled for an individual request, and M be the amount of accessible virtual machines (P = P1, P2,..., Pi,..., PM). Let I(Tj) denote the amount of instructions.

Let Sr(Pi) function as the computing average, that will be the machine's ability to calculate a million directives in a second, S(Pi) computing node's cost, Mc(Pi) be the memory usage cost, and BWc(Pi) be the expense of bandwidth usage for every single machine (Pi). The collection of processors (P) and tasks (T) can be represented mathematically using the following vectors: P = {P1, P2, P3,..., PM}, T = {T1, T2, T3,..., TN}. Also if Tij signifies that task j is treated by virtual machine i the solution can be given as follows:

$$Sol = \{T^i_1 , T^i_2 , T^i_3 , \dots, T^i_j , \dots, T^M_N \} \quad (3)$$

Using the mathematical model below, the task scheduling issue could be a stated multiobjective optimization problem:

$$Minimize : \sum_{i=1}^{M}\left(max\ 1 \le j \le N \left(\frac{I(T_j)}{Sr(p_i)}\right), \forall T_j \in p_i\right) \quad (4)$$

$$Minimize : \sum_{i=1}^{M}\left(\sum_{T_j \in P_i}\left(TotalCost\ (T_j^i)\right)\right) \quad (5)$$

where $\left(\frac{I(T_j)}{Sr(p_i)}\right)$ is the time it takes node i to complete task j, which might be calculated by divide the quantity of instructions in task j (I) by node i's processing unit rate (Sr), The overall computational cost of Pi, memory, and bandwidth resource utilized by node i to handle task j is calculated as TotalCost(Tij), and it can be calculated by adding the three combinations together:

$$TotalCost(T^i_j) = Sc(T^i_j) + Mc(T^i_j) + Bc(T^i_j) \quad (6)$$

The following three formulae can be used to compute each cost separately. The equations show the computational, RAM, and bandwidth expenses that node i will incur to complete task j.

$$S_c\,(T^i_j) = S_c(P_i) \times \left(\frac{I(T_j)}{Sr(p_i)}\right) \quad (7)$$

$$M_c(T^i_j) = M_c(P_i) \times RM(T_j) \quad (8)$$

$$B_c(T^i_j) = BWc(P_i) \times RB(T_j) \quad (9)$$

The first objective function in the model in Eq. (4), which reduces the overall response time required by way of a system to answer users' requests. Eq. (5) depicts the 2nd objective function, which seeks to reduce the full total cost of resource utilization.

4. Proposed work

4.1 Proposed model

This part discusses how an improved CS algorithm was used to create the proposed task scheduling model. Since continuous values generated by the Continuous CS algorithm cannot be allocated to appropriate computing nodes, the operators are ineffective for task scheduling in dynamic cloud environments. Levy Flight applied for continuous space. As a result, have been some changes to the Levy flight equation to solve this problem by searching in discrete space, as shown in Fig. 1.

4.2 Initial population and solution representation

The original population may be the collection of all individuals utilized by the CS to discover the best solution. Believe that the populace is N individuals. The N people are initialized randomly to find out many locations in the search space, in addition, to make sure the variety of the people in the initial generation. Individuals are chosen from the initial population and some operations are performed in it to produce the following generation. The solution is represented as a vector that includes the tasks to be performed by the available processors Fig. 2 depicts one possible candidate solution.

4.3 Fitness evaluation

After the generation of the initial solution, each individual's fitness value is evaluated and saved for future reference. Eqs. (4) and (5) can be used to define the fitness function. We employed a multi-objective function that took into consideration cost and makespan in this case. As shown in Eq. (4), the minimum fitness function value is set by the minimum cost, minimal makespan by used Eq. (5).

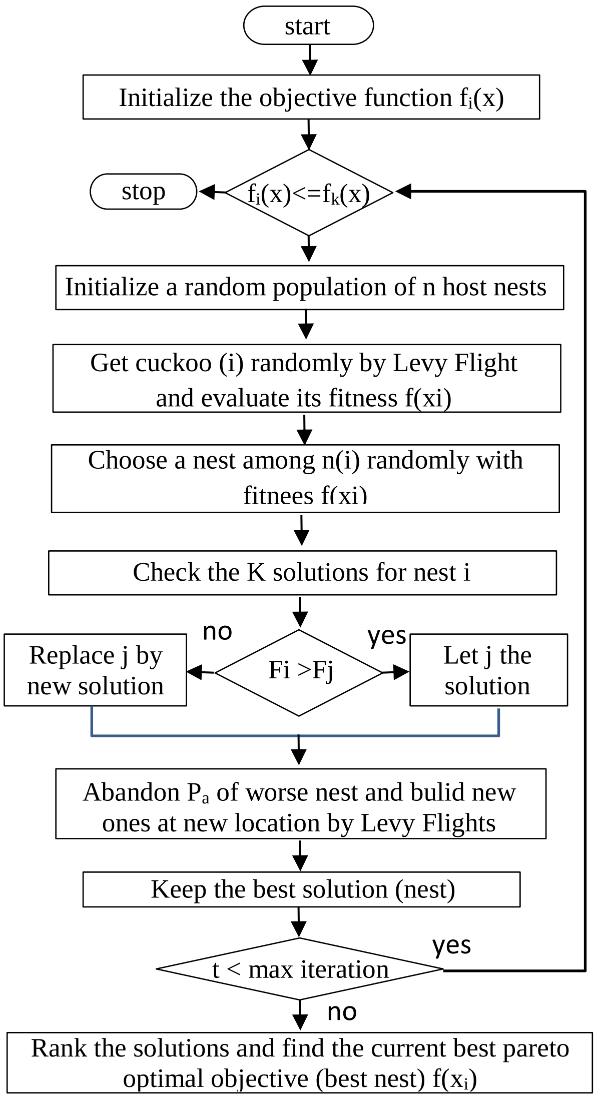
start
Initialize the objective function $f_i(x)$
$f_i(x)<=f_k(x)$
stop
Initialize a random population of n host nests
Get cuckoo (i) randomly by Levy Flight and evaluate its fitness f(xi)
Choose a nest among n(i) randomly with fitnees f(xi)
Check the K solutions for nest i
no
yes
Replace j by new solution
Fi >Fj
Let j the solution
Abandon P_a of worse nest and bulid new ones at new location by Levy Flights
Keep the best solution (nest)
yes
t < max iteration
no
Rank the solutions and find the current best pareto optimal objective (best nest) $f(x_i)$

Figure. 1 Digram of Multi-objective improved cuckoo search (MOICS) algorithm

Tasks	T1	T2	T3	T4	T5	T6	T7
Nodes	3	1	3	2	2	1	3

Figure. 2 Initial solution format

4.4 Levy flights

It has the potential to conduct a comprehensive search near the solution, which can be followed by a large footstep in the lengthy run. It's aforesaid that Levy flights are used to efficiently look for a new best solution in most optimization problems [31]. Levy Flight applied to continued space. As a result, have suggested some changes to the Levy flight equation to allow for discrete space searching.

4.5 Pareto optimum solutions

If none of the objective functions could be improved without worsening a few of the other objective values, the solution is recognized as Pareto optimum.

4.6 Multiobjective improved cuckoo search (MOICS) algorithm

CS algorithm uses the three idealized rules listed in Section 2.2 to deal with a single optimization function, changed only the 1st and 3rd rules to combine the criteria of multiobjective optimization with kth various objectives.

Mathematically, the first rule is randomly transformed to create a new, random solution using Levi's trip or random walk, where a random switch occurs on the solutions, while the second law remains the same in principle to ensure that the best solutions are provided to the next generation, and the third law is applied to the transformation process and in this way solutions are rejected the worst. The MOICS algorithm's efficiency is ensured by these one-of-a-kind functions.

- In the MOICS algorithm, the below parameters are utilized.
- Pa ∈ [0, 1] The probability that a bad nest is likely to be abandoned.
- $\alpha > 0$ step size, which should be related to the magnitude of the attention issue. In the vast majority of cases, $\alpha > 1$.
- λ random step length.

In the process of creating new solutions to replace old ones. The worst solutions were replaced by randomly generating solutions in the state space in the standard CS generating solution. This can make convergence to an optimal solution more difficult. wherefore, has been proposed splitting the abandoned nests into two parts, with the first half being replaced by random solution generation and the second being created by taking the vector that represents the solution and performing the mutation procedure as shown in Algorithm. 2 and Algorithm. 3, while the MOICS algorithm's flow is depicted in Algorithm. 1.

Algorithm 2. Multi-Objective Improved Cuckoo Search (MOICS)

```
Input: Population of the problem, $p_a$
Output: $S_{best}$
1   Initialize the objective functions $f_1(x), f_2(x) \dots F_K(x)$, $x = (x_1, \dots, x_d)^T$;
2   Initial a population of n host nests $x_i$ ($i = 1, 2, \dots, n$),
3   Probability $P_a \in [0,1]$ and Maximum number of iteration $Max_{itr}$;
4   while : (($t < Max_{itr}$) or (Stop Condition)) do
5       Get a Cuckoo (say i) randomly by Levy flights; // new solution $x_i^{(t+1)}$
6       Evaluate its quality/fitness $F_i$; // $F_i = f(x_i^{(t+1)})$
7       Choose a nest among n (say j) randomly; //old solution $x_i^t$
8       Evaluate the K solutions for nest j
9       if ($F_i > F_j$) then        // $x_i^{(t+1)} > x_i^t$
10          Replace $F_j \leftarrow F_i$; // old solution $x_i^t$ with new solution $x_i^{(t+1)}$
11      end if
12      A percentage (Pa)/2 of the worst nests are abandoned, and new ones are constructed at random;
13      A fraction $(P_a)/2$ of worse nests are abandoned and new
14      ones are built by call mutation procedure
15      Keep the best solutions (or nests with quality solutions);
16      Rank the solutions and find the current best Pareto optimal solutions;
17      $t \leftarrow t+1$;
18  end while
19  return $S_{best}$;
end
```

To maintain robust randomization in exploring the country space, we held a random generation of solutions replacing abandoned solutions.

In single optimization situations where one egg occurs in a nest with regards to the worth of the objective function, ranking the nests on the basis of the quality of the solution is simple. However, ranking the nests is a substantial task in multi-objective situations with multiple eggs in each nest. The strategy of contrasting the solutions is dependant on objective function principles, which will be wrong and results in an incorrect evaluation. It could happen if your objectives are in dispute with each other. Nests are separated into two classes using Pareto dominance to accomplish this. Non-dominated nests with Pareto optimal solutions are of interest to the first collection, while dominated nests with non-Pareto optimal dominated solutions are of interest to the second set.

Algorithm 3. Generate a new solution from an abandoned nest

```
Input: D : (pa)/2.
Output: New Nests.
1   for i=1 to D
2       Generate a new solution by call mutation procedure
3   End for
```

5. Result and comparison

The simulation setup and evaluation of the proposed algorithm are described in the following sections based on the results of the conducted experiments.

5.1 Bee life algorithm (BLA)

As a novel meta-heuristic approach, the artificial bee life algorithm [32] was created in 2005.Modeling honey bees' smart foraging behavior in a search process for solving real-parameter optimization problems [33] is based in part on the hybrid foraging behavior of honey bees.

By using the bee swarm algorithm, in [18] authors improve task scheduling utilizing fog computing. The goal is to find the right balance between CPU execution time and allocated memory.

5.2 Round robin algorithm

In round robin, regardless of the load on the VM, the next VM in queue gets assigned tasks.Neither resource capabilities, priorities, nor task duration is taken into account in the Round Robin strategy.Due to this, higher priority tasks and longer tasks require longer response times [34, 35].

5.3 Modified particle swarm optimization

An evolutionary algorithm, known as particle swarm optimization, developed by Kennedy and Eberhart [36], simulates the collective behavior of flocks of birds or groups of fish taking a course towards their destination. In modified particle swarm optimization (MPSO) [10], which can adapt to the proposed model, this is accomplished by adding or removing layers to the PSO algorithm while maintaining the concept of PSO. Based on the best experment (pBest) and leader (gBest), particle velocities are updated daily.

5.4 Experimental settings

The tests in this study were conducted out on a PC running Windows 10, with a 2.8 GHz Core i7 processor and 16 GB of RAM, and using Python 3.8.5 software. Has been solved 11 separate task-scheduling problems with varying numbers of tasks to assess the efficiency of the proposed MOICS (40 to 500 tasks). The computing power and resource consumption costs of cloud nodes are different. Each node is presumed to have its processing power, as calculated by MIPS (million instructions per second), as well as Processor, memory, and bandwidth usage costs. The Cloud system was built with thirteen processing nodes, which have the characteristics described in Table 2. Servers or virtual computers in high-performance data centers undertake tasks at the Cloud tier. As a result, Cloud nodes process data significantly more quickly. The price of consuming energy in the Cloud is higher, and these charges are measured in Grid Dollars (G$) [23], a currency unit utilized in the simulation to substitute actual money.

Table 2. The cloud infrastructure's characteristics

Parameters	Cloud tier	Unit
Number of nodes	13	node
CPU usage cost	[0.1, 1.0]	G$/s
Memory usage cost	[0.01, 0.05]	G$/MB
Bandwidth usage cost	[0.01, 0.1]	G$/MB
CPU rate	[500, 5000]	MIPS

Table 3. Attributes of tasks

Property	Value	Unit
Input file size	[10, 100]	MB
Memory required	[50, 200]	MB
Output file size	[10, 100]	MB
Number of instructions	[1, 100]	10^9 instructions

All user queries are routed through the cloud system. Each request is dissected into numerous tasks, which are then assessed and the resources required determined. The quantity of memory required the amount of instructions, how big is the I/O file are all assumed to be features of each task. With regards to the demand, the number of tasks directed at each request may differ significantly. The attributes listed in Table 3 were used to define the roles for every dataset at random. The experiment may cover numerous situations, with some requiring plenty of computing and others demanding more bandwidth utilization or memory because several distinct forms of jobs were constructed.

5.5 Experimental results

As part of this analysis, we will assess how well MOICS performs in a cloud computing environment by solving the task scheduling problem. Table 4 summarizes the parameter settings that have an impact on algorithm execution.

In Table 5 and 6, the results of MOICS, MPSO, BLA, TCAS and RR for 3 systems are shown, each with ten runs. Each run includes tasks m=[40, 80, 120, 160, 200, 250, 300, 350, 400, 450, 500] and the cloud system model with nC=13. Competitie results are highlighted in bold. The makespan function is shown in Table 5, while the cost function is shown in Table 6.

Table 4. Settings used for testing the efficiency of the proposed algorithms

Parameter name	Acronym	Possible settings
Number of systems	nSystem	10
Number of Runs	nRun	3
Number of node	nC	13
Number of Tasks	m	{40,80,120,160,200,250, 300,350,400,450,500}
Population Size	I	100
Number of Generation	Max_g	500

Table 5. Comparison of the five algorithms' makespan and overall cost

Makespan					
No. of Task	MOICS	TCAS*	BLA*	MPSO*	RR*
40	**151.461**	191.44	207.57	215.27	456.11
80	**337.907**	394.69	416.09	445.35	963.08
120	**489.144**	607.71	654.92	686.2	1495.66
160	**724.244**	819.97	917.67	932.33	1912.08
200	**906.207**	940.87	1067.49	1065.94	1905.7
250	**1137.66**	1270.96	1490.65	1479.84	3054.8
300	**1388.24**	1473.79	1765.49	1712.76	3309.14
350	**1642.56**	1949.04	2010.74	1911.27	3638.19
400	**1829.65**	1944.58	2421.98	2300.51	4347.64
450	**2177.56**	2235.16	2840.79	2751.29	5350.35
500	**2371.17**	2503.09	3174.39	3067.26	6023.74

* Results of these algorithms from the research paper for Nguyen et al [23].

Table 6. Comparison of the five algorithms' overall cost

Cost					
No. of Task	MOICS	TCAS*	BLA*	MPSO*	RR*
40	**605.244**	733.85	730.88	740.38	755.98
80	**1377.27**	1540.23	1540.85	1533.4	1581.82
120	**2052.76**	2363.37	2367.59	2381.2	2418.9
160	**2927.76**	3176.17	3183.8	3197	3246.69
200	**3692.46**	3755.21	3767.37	3778.3	3835.4
250	**4617.60**	4988.94	5017.17	5007.6	5079.06
300	**5657.33**	5832.69	5877.41	5862.5	5935.94
350	**6604.98**	6607.52	6653.37	6632.4	6738.39
400	**7473.85**	7738.56	7816.04	7760	7875.39
450	**8673.28**	8845.9	8926.57	8876.3	9016.59
500	**9538.44**	9902.64	9995.97	9921.8	10097.7

* Results of these algorithms from the research paper for Nguyen et al [23].

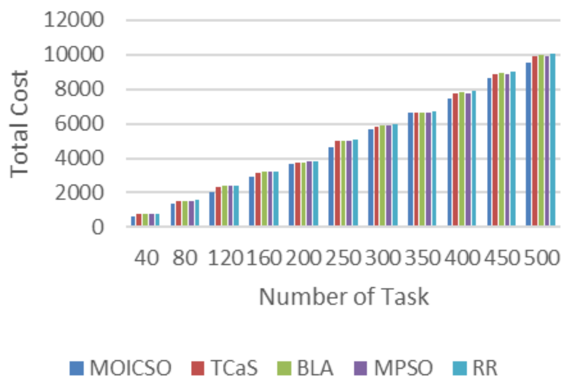

Figure. 3 Total cost comparison of the five algorithms

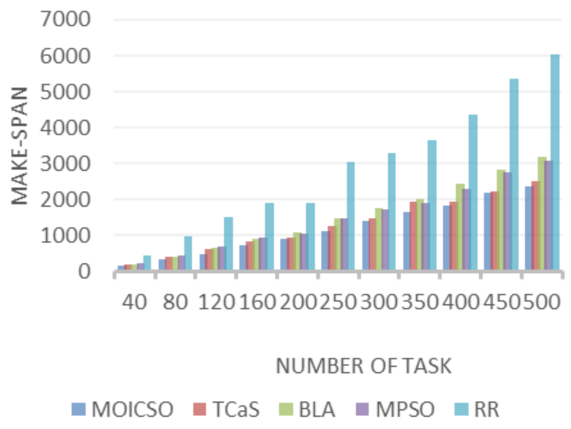

Figure. 4 Make-Span comparison of the five algorithms

Total–cost for five strategies is compared in Fig. 3. Every dataset with a high average fitness showed that our suggested algorithm, MOICS, dominated the first place. Meanwhile, Fig. 4 compares the Makespan of the proposed model MOICS algorithm to the four others while the number of tasks changed, demonstrating that our suggested MOICS algorithm outperforms the others. Based on the results, our suggested method, MOICS, could achieve the best barter among make-span and overall cost than the other four algorithms, as well as demonstrating the supremacy of time optimization.

6. Conclusion and future work

In this work, we addressed tasks scheduling problems for BoT applications in cloud computing environments. The MOICS algorithm can solve the problem as a multi-objective optimization problem to reach a good trade-off between cost and execution times when completing a set of tasks in a Cloud system. In this case, swarm algorithms are used to solve the problem. In order to solve the problem, a swarm algorithms is adopted. In a comparison of 11 sets of tasks in a Cloud system, MOICS performed better than the MPSO, TCAS, BLA, and RR methods in terms of the trade-off between makespan and cost execution.

In the foreseeable future more algorithms, particularly metaheuristic algorithms will be researched, improved, and used to solve scheduling problems. We also intend to broaden the scope of the scheduling issues by focuses on a variety of other objectives, like waiting time, throughput, and energy consumption, to meet the needs of users. For greater practicality, budget, deadline, and resource constraints can be added.

Conflicts of Interest

No conflict of interest.

Author Contributions

The paper conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualization, have been done by 1st author. The supervision and project administration, have been done by 2nd and 3rd authors.

References

[1] X. Q. Pham, N. D. Man, N. D. T. Tri, N. Q. Thai, and E. N. Huh, "A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing", *International Journal of Distributed Sensor Networks*, Vol. 13, No. 11, p. 1550147717742073, 2017.

[2] A. T. S. A. Obaidi and S. A. Hussein, "Two improved cuckoo search algorithms for solving the flexible job-shop scheduling problem", *International Journal on Perceptive and Cognitive Computing.*, Vol. 2, No. 2, 2016.

[3] A. T. S. A. Obaidi and H. S. Abdullah, "Camel herds algorithm: A new swarm intelligent algorithm to solve optimization problems", *International Journal on Perceptive and Cognitive Computing.*, Vol. 3, No. 1, 2017.

[4] S. Basu, M. Karuppiah, K. Selvakumar, K. C. Li, S. H. Islam, M. M. Hassan, and M. Z. A. Bhuiyan, "An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment", *Future Generation Computer Systems.*, Vol. 88, pp. 254-261, 2018.

[5] D. Chahal, B. Mathew, and M. Nambiar, "Simulation based job scheduling optimization for batch workloads", In: *Proc. of the 2019 ACM/SPEC International Conf on Performance Engineering*, pp. 313-320, 2019.

[6] D. Rahbari and M. Nickray, "Scheduling of fog networks with optimized knapsack by symbiotic organisms search", In: *Proc .of 2017 21st Conference of Open Innovations Association*, pp. 278-283, 2017.

[7] D. Vasiljevic, "Classical and evolutionary algorithms in the optimization of optical systems", *Springer Science & Business Media*, 2012.

[8] N. Soltani, B. Soleimani, and B. Barekatain, "Heuristic Algorithms for Task Scheduling in Cloud Computing: A Survey.", *International Journal of Computer Network & Information Security.*, Vol. 9, No. 8, 2017.

[9] P. Sun, Z. Cai, and D. Liu, "Budget Constraint Bag-of-Task Based Workflow Scheduling in Public Clouds", In: *Proc. of CCF Conference on Computer Supported Cooperative Work and Social Computing*, pp. 243-260, 2019.

[10] S. Abdi, S. A. Motamedi, and S. Sharifian, "Task scheduling using modified PSO algorithm in cloud computing environment", In: *Proc. of International Conference on Machine Learning, Electrical and Mechanical Engineering*, Vol. 4, No. 1, pp. 8-12, 2014.

[11] R. K. Jena, "Multi objective task scheduling in cloud environment using nested PSO framework", *Procedia Computer Science.*, Vol. 57, pp. 1219-1227, 2015.

[12] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. S. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing", *IEEE Transactions on Cloud Computing*, 2019.

[13] L. Liu, D. Qi, N. Zhou, and Y. Wu, "A task scheduling algorithm based on classification mining in fog computing environment", *Wireless Communications and Mobile Computing.*, Vol. 2018, 2018.

[14] C. Tang, "A mobile cloud based scheduling strategy for industrial internet of things", *IEEE Access*, Vol. 6, pp. 7262-7275, 2018.

[15] C. W. Tsai, W. C. Huang, M. H. Chiang, M. C. Chiang, and C. S. Yang, "A hyper-heuristic scheduling algorithm for cloud", *IEEE Transactions on Cloud Computing.*, Vol. 2, No. 2, pp. 236-250, 2014.

[16] R. Z. Naeem, S. Bashir, M. F. Amjad, H. Abbas, and H. Afzal, "Fog computing in internet of things: Practical applications and future directions", *Peer-to-Peer Networking and Applications.*, Vol. 12, No. 5, pp. 1236-1262, 2019.

[17] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for

achieving optimal power-delay tradeoff in edge cloud systems", In: *Proc. of 2016 IEEE International Conference on Communications*, pp. 1-7, 2016.

[18] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm", *Enterprise Information Systems.*, Vol. 12, No. 4, pp. 373-397, 2018.

[19] P. Krishnadoss, N. Pradeep, J. Ali, M. Nanjappan, P. Krishnamoorthy, and V. K. Poornachary, "CCSA: Hybrid cuckoo crow search algorithm for task scheduling in cloud computing", *International Journal of Intelligent Engineering and Systems.*, Vol. 14, No. 4, pp. 241-250, 2021.

[20] J. Xu, Z. Hao, R. Zhang, and X. Sun, "A method based on the combination of laxity and ant colony system for cloud-fog task scheduling", *IEEE Access*, Vol. 7, pp. 116218-116226, 2019.

[21] H. R. Boveiri, R. Khayami, M. Elhoseny, and M. Gunasekaran, "An efficient Swarm-Intelligence approach for task scheduling in cloud-based internet of things applications", *Journal of Ambient Intelligence and Humanized Computing.*, Vol. 10, No. 9, pp. 3469-3479, 2019.

[22] S. Kodli and S. Terdal, "Hybrid max-min genetic algorithm for load balancing and task scheduling in cloud environment", *International Journal of Intelligent Engineering and Systems.*, Vol. 14, No. 1, pp. 63-71, 2021.

[23] B. M. Nguyen, H. T. T. Binh, and B. D. Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud-fog computing environment", *Applied Sciences.*, Vol. 9, No. 9, p. 1730, 2019.

[24] A. M. S. Kumar and M. Venkatesan, "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment", *Wireless Personal Communications.*, Vol. 107, No. 4, pp. 1835-1848, 2019.

[25] W. Wu, H. R. Maier, and A. R. Simpson, "Single-objective versus multiobjective optimization of water distribution systems accounting for greenhouse gas emissions by carbon pricing", *Journal of Water Resources Planning and Management.*, Vol. 136, No. 5, pp. 555-565, 2010.

[26] Y. Sun, F. Lin, and H. Xu, "Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II", *Wireless Personal Communications.*, Vol. 102, No. 2, pp. 1369-1385, 2018.

[27] Y. Chen, J. Huang, C. Lin, and X. Shen, "Multi-objective service composition with QoS dependencies", *IEEE Transactions on Cloud Computing.*, Vol. 7, No. 2, pp. 537-552, 2016.

[28] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights", In: *Proc. of 2009 World Congress on Nature & Biologically Inspired Computing*, pp. 210-214, 2009.

[29] H. Zheng and Y. Zhou, "A novel cuckoo search optimization algorithm based on Gauss distribution", *Journal of Computational Information Systems.*, Vol. 8, No. 10, pp. 4193-4200, 2012.

[30] B. T. B. Khoo, B. Veeravalli, T. Hung, and C. W. S. See, "A multi-dimensional scheduling scheme in a Grid computing environment", *Journal of Parallel and Distributed Computing.*, Vol. 67, No. 6, pp. 659-673, 2007.

[31] A. F. Kamaruzaman, A. M. Zain, S. M. Yusuf, and A. Udin, "Levy flight algorithm for optimization problems-a literature review", *In: Applied Mechanics and Materials. Trans Tech Publications Ltd*, Vol. 421, pp. 496-501, 2013.

[32] D. Karaboga, "An idea based on honey bee swarm for numerical optimization", *Technical Report-tr06*, *Erciyes University, Engineering Faculty, Computer Engineering Department*, 2005.

[33] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", *Journal of Global Optimization.*, Vol. 39, No. 3, pp. 459-471, 2007.

[34] D. C. Devi and V. R. Uthariaraj, "Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks", *The Scientific World Journal.*, Vol. 2016, 2016.

[35] P. Pradhan, P. K. Behera, and B. N. B. Ray, "Modified round robin algorithm for resource allocation in cloud computing", *Procedia Computer Science.*, Vol. 85, pp. 878-890, 2016.

[36] J. Kennedy and R. Eberhart, "Particle swarm optimization", In: *Proc. of ICNN'95-International Conf, on Neural Networks*, Vol. 4, pp. 1942-1948, 1995.