DOI: https://doi.org/10.33103/uot.ijccce.22.1.9

# Task Scheduling in Cloud Computing Based on The Cuckoo Search Algorithm

Sajjad Jaber[1], Yossra Ali[2], Nuha Ibrahim[3]
[1,2,3]*Department of Computer Science, University of Technology, Baghdad, Iraq*
[1]*sajjad.sh.jaber@gmail.com,* [2]*110017@uotechnology.edu.iq,* [3]*110009@uotechnology.edu.iq*

***Abstract****— Task scheduling is one of the very crucial facets of cloud computing. The task scheduling method must assign jobs to virtual machines. In cloud computing, task scheduling includes a frontal influence on a system's resource utilization and operational costs. Diverse meta-heuristic algorithms, in addition to their modifications, have been developed to improve the efficiency of task executions in the cloud. In this paper, a multiobjective optimization model is applied using the metaheuristics cuckoo search optimization algorithm (MCSO) to enhance the performance of a cloud system with limited computing resources while minimizing the time and cost. Finally, we analyze the performance of the proposed MCSO with the existing methods, such as Bee Life Algorithm (BLA), A Time–Cost aware Scheduling (TCaS) algorithm, Modified Particle Swarm Optimization (MPSO), and Round Robin (RR), for the evaluation metrics makespan and cost. Based on the outcomes of the experiments, it can be inferred that the proposed MCSO provides essential schedule jobs with the shortest makespan and average cost.*

***Index Terms****— Task Scheduling, Multiobjective, Cloud Computing, Cost, Makespan.*

## I. INTRODUCTION

Cloud computing is becoming more common in industry, academia, and society as access to the Internet and big data rise in volume, velocity, and variety over the Internet. Distributed computing, utility computing, grid computing, and autonomous computing all fall under the umbrella of cloud computing [1]. It's a new model that uses the Internet to deliver software, infrastructure, databases, security, platforms, storage, computing, and hardware as services. Computing entities are virtualized, dynamically configurable, and driven by economic scale in a cloud computing environment [2]. Recently, researchers are becoming enthusiastic about the topic of task scheduling in a distributed system. 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. A bi-objective optimization challenge is finding the best balance between resolution time and makespan in a precedence-constrained concurrent program [3]. Scheduling is a way of making decisions and provides 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 [4], [5]. Scheduling in the cloud entails mapping n independent/dependent tasks to m resources to be able to meet a set of user-defined or system-defined requirements. 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. All non-dominated solutions constitute the Pareto-optimal front [6], [7].

*Received 3/August/2021; Accepted 18/October/2021*
 *ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.1.9

Since the cloud scheduling issue is Non-deterministic Polynomial-time hard (NP-hard), finding optimal scheduling solutions in polynomial time is impossible [8]. If such issues are solved utilizing an exhaustive search, the time it will take to generate a scheduling solution might be exceedingly long. Several research projects have been launched to get near-optimal scheduling solutions using evolutionary and swarm intelligence-based algorithms. Through the scheduling process, two or more opposite objectives are examined for optimization in multi-objective scheduling [9]. Several optimization scheduling issues need the optimization of more than one objective function. There is no single solution to such issues. Rather, good barter solutions that act the most effective potential compromises between the scheduling criteria or objectives can be identified [10]. Many factors should be considered to formulate the multi-objective optimization problem, such as a system to maintain non-dominated solutions over populations, a technique to maintain population variety, and a mechanism to lead the search to the Pareto-optimal front.

By using the cuckoo search optimization algorithm, we introduced a Multi-objective Cuckoo Search Optimization (MCSO) based task scheduling in this study. The makespan and cost multi-objective functions are utilized in this paper. We get the near-optimal scheduled task using the multi-objective function. Our approach was evaluated and compared to the Modified Particle Swarm Optimization (MPSO) algorithm, Bee Life Algorithm (BLA) algorithm, TCaS algorithm, and RR algorithm on a variety of datasets of various sizes. The findings show that the proposed algorithm provide the optimal Quality of Service while also being faster and less expensive than the other options. The following are the significant contributions to task scheduling research:

- For task scheduling, an approach known as MCSO is used, which includes the benefits of swiftly converging and simply realization, provide a near-optimal solution.
- Multiobjective task scheduling problems are formulated for cloud computing which intended to minimize makespan and cost in the cloud environment.

The following sections of the paper are organized as follows: Section 2 discusses a review of related work on existing task scheduling techniques. The definition of the task scheduling problem is presented in Section 3. Section 4 discusses the methodology. The results and comparison are described in Section 5. The conclusion is found in section 6.

## II. RELATED WORK

Several scheduling approaches for cloud computing are presented in the curent state of art. For creating near-optimal schedules, Pan et al. [11] suggested a new artificial chemical reaction optimization approach. The objects react with each other inside their chemical reaction process to achieve the lowest possible energy case, which optimizes grid scheduling's makespan. The proposed method is compared to the Gentic Algorithm (GA) algorithm and the Heterogeneous Earliest Finish Time Algorithm (HEFT). While Salimi et al. [12] presented Non-dominated Sorting Genetic Algorithm (NSGA-II), which uses a fuzzy variance-based crossover technique to maximize two scheduling objectives: resource use cost and makespan. In [13], they introduced multi-objective task scheduling utilizing a cuckoo and particle swarm optimization technique. The major goal of this work is to reduce the time, cost, and rate of deadline violations, to get the near-optimal task scheduling using the multi-objective function. Mohamed Abd Elaziz et al. [14] have proposed the Moth Search Differential Evolution (MSDE) algorithm as an alternate way for solving the task scheduling issue in a cloud computing milieu, which combines the Moth Search Algorithm

Received 3/August/2021; Accepted 18/October/2021
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.1.9

(MSA) and Differential Evolution (DE). They conclusioned that the suggested MSDE can efficiently schedule tasks to the Virtual Machine (VM) while consuming the least amount of time. Whereas V. M. Arul Xavier and S. Annadurai. [15] handle the challenge of job scheduling in heterogeneous virtual computers, a chaotic social spider algorithm based on a social spider. This work focuses on minimizing the overall makespan with proper load balancing by modeling the swarm intelligence of social spiders utilizing chaotic inertia weight-based random selection. A work scheduling strategy for cloud-based on the multiobjective Artificial Bee Colony Algorithm (TA-ABC) is presented in [16]. The suggested algorithm optimizes the cloud environment's cost, resource utilization, processing time, and energy. Mohit Agarwal and Gur Mauj Saran Srivastava [17] introduced a meta-heuristic technique Genetic Algorithm enabled Particle Swarm Optimization (PSOGA). In this work the problem of task scheduling in a cloud computing milieu, of hybrid version of the PSO and GA algorithms is presented. PSOGA reduces makespan time by combining the PSO's variegation and GA's condensation properties, and the approach has only been tested on small datasets. In [18] a new approach was described to the prognosis of Tasks Computation Time. using Principle Components Analysis (PCA) and diminishing the Expected Time to Compute (ETC) matrix, the suggested mechensim improves the makespan and decreases computation and complexity. According to Xu et al [19] , the Min-min-based time and cost barter (MTCT) was a multi-objective heuristic approach for optimizing time and cost metrics with fault-tolerant considerations, based on particle swarm optimization (PSO) technique.[20] presented cloud task scheduling based on a multi-objective model and the Grey Wolf Optimization (GWO) algorithm to reduce cost and time in cloud environments. A multiobjective scheduling strategy for hybrid clouds with the goal of reducing Makespan and Cost was presented by Zhou et al [21]. Bitam et al. [22] suggested the Bee Life Algorithm (BLA) as a task scheduling technique. The emphasis of the study is on key goals: memory and execution time, and the approach has only been tested on small datasets. In a Cloud–Fog computing scenario, Binh et al [23] introduced a task scheduling technique based on a Genetic Algorithm (GA). This strategy seeks to strike a good time-cost balance. The proposed technique outperforms the Bee Life approach; however, because the algorithm was only evaluated on short datasets, the tests were limited.

Most existing task scheduling algorithms, as previously indicated, are useful for short datasets and only evaluate cost or makespan as a single aim. Although some trade-off study has been done previously, those tradeoffs are unlikely to truly reflect reality. When coping with two or more competing goals, Pareto optimization approaches, also referred to as multi optimization algorithms, are crucial. Kumar and Venkatesan [24] proposed a hybrid genetic-ACO algorithm to solve a multiobjective task scheduling approach to tackle a multiobjective task scheduling issue. Multiobjective problems are more practical than single-objective issues in today's task scheduling systems [25], [26], [27]. To make the better decision suitable, the multi-objective issue necessitates a barter between numerous goals. As a result, this research points to a problem with multi-objective task scheduling in cloud environments. To accomplish so, we developed the MOCS algorithm, which is capable of effectively addressing multi-objective optimization problems.

## III. FORMULATION AND DEFINITION OF PROBLEM

Consider the case where there are n tasks [T1, T2, T3,... Tn] and m processing machines [M1, M2, M3,... Mm] which are obtainable to compute those tasks that must fulfill the situation m < n, that is, the numeral of tasks should be more than the number of processing or Virtual Machines (VMs),if there is no interdependence between tasks, they

Received 3/August/2021; Accepted 18/October/2021
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.1.9

could be performed in just about any order. The task length is expressed in MI (Million Instructions) and the useful resource ability is measured in MIPS (Million Instructions Per Second). By dividing the job length by the resource capacity, the Expected Time to Complete (ECT) of each task on a single processor is computed. The goal is to schedule the supplied tasks on VMs to obtain the lowest cost, shortest time to completion, and highest VM usage. Table I lists the notations for the majority of mathematical symbols. The mathematical model we used in this research is based on [28].

TABLE I. THE SYMBOLS USED IN THE WORK

| Symbol | Description |
|---|---|
| $P_i$ | number machine i |
| $T_j$ | number task j |
| $T_i^j$ | task j is handled by machine i |
| $I(T_j)$ | total number of j task instructions |
| $RB(T_j)$ | the desired bandwidth for task j |
| $RM(T_j)$ | desired 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 |

### A. Cost

When a task is completed in the cloud system, a fee is charged to cover processing costs ($S_c$), memory costs($M_c$), bandwidth costs ($BW_c$), and all other procsessing cost. The following is how the total cost of all tasks to be completed in the Cloud system is calculated:

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

$$\text{TotalCost(Tij)} = \text{Sc (Tij)} + \text{Mc (Tij)} + \text{Bc (Tij)} \quad (2)$$

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.

$$S_c\left(T^i_j\right) = S_c(P_i) \times \left(\frac{I(T_j)}{S_r(P_i)}\right) \quad (3)$$

$$M_c\left(T^i_j\right) = M_c(P_i) \times RM(T_j) \quad (4)$$

$$B_c\left(T^i_j\right) = BWc(P_i) \times RB(T_j) \quad (5)$$

### B. Makespan

The maximum time wanted by tasks in a specific schedule to complete their execution is known as makespan. We used Eq. (6) to get the ETC (Expected Time to Complete) for each task.

$$ETC = \begin{bmatrix} ETC_{1,1} & ETC_{1,2} & ETC_{1,3} & \cdots & ETC_{1,N_{vm}} \\ ETC_{2,1} & ETC_{2,2} & ETC_{2,3} & \cdots & ETC_{2,N_{vm}} \\ ETC_{3,1} & ETC_{3,2} & ETC_{3,3} & \cdots & ETC_{3,N_{vm}} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ ETC_{N_{tsk},1} & ETC_{N_{tsk},2} & ETC_{N_{tsk},3} & \cdots & ETC_{N_{tsk}N_{vm}} \end{bmatrix}$$

$$\text{Minimize: } \sum_{i=1}^{M}\left(\max 1 \le j \le N\left(\frac{I(T_j)}{S_r(P_i)}\right),\ \forall\, T_j \in P_i\right) \quad (6)$$

Where $\left(\frac{I(T_j)}{S_r(P_i)}\right)$ is the time taken for 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).

*Received 3/August/2021; Accepted 18/October/2021*
 *ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.1.9

The scheduling problem has been reformulated as a multi-objective optimization problem, in which two objectives are optimized at the same time, where the makespan and total cost are reduced. In each scenario, both functions are prioritized and minimized at the same time. For each scenario, there's no single ideal answer, but instead, some viable solutions that are all optimal in some ways. The distance metric measures how close an algorithm's non-dominated solutions are to the reference Pareto-optimal front. The box plots and spread measure provide information on solution variety and distribution in the search space.

## IV. METHODOLOGY

In this part, the Cuckoo Search (CS) algorithm is discribed for individual optimization and multi-objective optimization to handle the issue of job scheduling in a cloud computing milieu. The following sections include the preface to cuckoo conduct, the efficiency of Levy flight, and the debate of the MCSO algorithm.

### A. Basic Terminologies

Cuckoo Search (CS) algorithm is a swarm-intelligence-based algorithm [29]. The naturalist conduct of cuckoos was used to create this algorithm. Every egg in a den represents a possible solution in this method. Generally, each cuckoo can only put one egg in a nest with a unique shape, yet each nest can contain multiple eggs representing a variety of solutions as shown in Algorithm.1. CS's major goal is to develop new solutions to replace the present population's worst solutions. Below is a step-by-step guide to the process.

Step 1 (Initialization): In this portion, a nest (solution) inhabitance is generated at random (Si, where I = 1,2,...n).

Step 2 (Fitness ssessment): Calculate the fitness function after generating solutions and then select the best one.

$$\text{fitness} = \text{minimum objective functions} \tag{7}$$

Step 3 (Creating a New Cuckoo): In this phase, we'll use levy flights to create new cuckoos. The objective function of each solution is then calculated to determine the quality of the solutions.

Step 4 (Updating): After calculating fitness, Eq.(8) is used to build a new solution.

$$I_i^{New} = I_i^{(t+1)} = I_i^t + \propto \oplus Levy(\lambda) \tag{8}$$

where $I_i^t$ denotes an older solution and $\oplus$ denotes entry-wise multiplication. A random walk with a random step size $\propto > 0$ following a levy distribution is known as levy flight.

$$\text{Levy u} = t^{-\lambda}\ (1 < \lambda \leq 3) \tag{9}$$

Step 5 (Reject the worst-case scenario): We reject the worst fitness value solution after the update process. The best solution is thought to be the best option.

Step 6 (Criterion stage Comes to an End): This method is repeated until the maximum iteration is reached. For further processing, the best option is picked.

we modified the original algorithm so that we can convert it from continuous to discrete to match the representation of the scheduling problem in the cloud computing system as shown in the next part.

**Algorithm 1.** Cuckoo Search Algorithm

1: **begin**
2: Objective function f(x), x=( $x_1$,…., $x_n$)$^T$;
3: Initial a population of n host nests $x_i$(i= 1,2,….,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 $F_i$

Received 3/August/2021; Accepted 18/October/2021
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.1.9

```
7:      Choose a nest among n (say j) randomly;
8:      if (Fi > Fj)
9:          Replace j with a new solution;
10:     end
11:     Relinquish a fraction (Pa) of worst nests;
12:     Keep the better 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
```

Where f(x) is fitness function, $x_i$ denoted to population nest, $F_i$ old fitness, and $F_j$ new fitness

**B. Multi-objective Cuckoo Search Optimization (MCSO) Algorithm**

The original CS algorithm deals with individual optimization functions and utilizes the 3 idealized rules:

- Every cuckoo place one ovum at a time, which it deposits in a nest that is picked at random.
- The better nest with the finest fineness eggs will be passed down to the following generation.
- The number of reachable steward nests is firm, and a host has a chance of discovering an unusual egg (0 or 1). In this instance, the steward has the option of throwing the egg or abandoning the nest and starting over in a new place.

Only the first and third rules are adjusted to include the multi-objective criteria for multi-objective 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 MCSO algorithm's efficiency is ensured by these one-of-a-kind functions. In the MOCS algorithm, the below parameters are utilized.

- $Pa \in [0, 1]$ 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, $\alpha > 1$.
- $\lambda$ 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. Therefore, it was used to create the proposed task scheduling model. Since continuous values generated by the CS algorithm cannot be allocated to appropriate computing nodes, the operators are ineffective for task scheduling in dynamic cloud environments. Levy Flight is applied for continuous space. As a result, there have been some changes to the Levy flight equation to solve this problem by searching in discrete space, as shown in Algorithm.2.

**Algorithm 2.** Multi-Objective Cuckoo Search Optimization (MCSO)

**Input:** Population of the problem, $p_a$
**Output:** $S_{best}$

1 Initialize the objective functions $f_1(x), f_2(x) \ldots F_k(x)$, $x = (x_1, \ldots, x_d)^T$;
2 Initial a population of **n** host nests $x_i$ (i = 1, 2, ..., n),

*Received 3/August/2021; Accepted 18/October/2021*
 *ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.1.9

```
Probability Pa ∈ [0,1] and Maximum number of iteration Max_itr;
while : ((t < Max_itr) or (Stop Condition)) do
    Get a Cuckoo (say i) randomly by Levy flights; // new solution x_i^(t+1)
    Evaluate its quality/fitness F_i; // F_i = f(x_i^(t+1))
    Choose a nest among n (say j) randomly; //old solution x_i^t
    Evaluate the K solutions for nest j
    if (F_i > F_j) then          // x_i^(t+1) > x_i^t
        Replace Fj ← F_i; // old solution x_i^t with new solution x_i^(t+1)
    end if
    Relinquish a fraction (Pa) of worst nests and build new ones at new locations via Levy
    flights;
    Keep the better solutions (or nests with quality solutions);
    Rank the solutions and find the current best Pareto optimal solutions;
    t← t+1;
end while
return S_best;
end
```

In single optimization situations where one egg occurs in a nest with regards to the worth of the objective function, ranking the nests based on 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.

## V. RESULT AND COMPARISON

The tests in this study were conducted on a PC running Windows Ten, with a 2.8 GHz Core i7 processor and 16 GB of RAM. Python is used to code the proposed algorithm. Different task (11) separate task-scheduling problems with varying numbers have been solved to assess the efficiency of the proposed MCSO (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 memory, and bandwidth usage costs. The cloud system was built with thirteen processing nodes, which have the characteristics described in Table II. 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. These costs are calculated according to Grid Dollars (G$)— a currency unit used in the simulation to substitute for real money

TABLE II. THE CLOUD INFRASTRUCTURE'S CHARACTERISTICS[28]

| 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 |

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 number of instructions, how big is the I/O file are all

*Received 3/August/2021; Accepted 18/October/2021*
 *ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.1.9

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 III 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. The characteristics used in Tables II and III depend on the research paper [28]

TABLE III. ATTRIBUTES OF TASKS[28]

| 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 |

We compared our approach to the BLA algorithm [19], MPSO algorithm [30], and TCaS algorithm [25], as well as RR algorithm [31]. The parameter settings for these methods are obtained from their original paper, and the halting condition (maximum number of iterations) is defined at 500 generations for all of them, including the proposed approach. They also ran 30 times to have better results, and Table IV shows the scheduling tallness and treatment cost of the five algorithms. Two key components contributed to the fitness function: makespan and total expenses. As a result of its high fitness value, the MOCS algorithm outperformed the competition on most all datasets in terms of both cost and time.

TABLE IV. COMPARISON OF THE FIVE ALGORITHMS' MAKESPAN AND OVERALL COST.

| **Makespan** | | | | | | **Cost** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **No. Of Task** | **MCSO** | **TCaS** | **BLA** | **MPSO** | **RR** | **MCSO** | **TCaS** | **BLA** | **MPSO** | **RR** |
| **40** | **186.385** | 191.44 | 207.57 | 215.27 | 456.11 | **666.364** | 733.85 | 730.88 | 740.38 | 755.98 |
| **80** | **374.022** | 394.69 | 416.09 | 445.35 | 963.08 | **1486.427** | 1540.23 | 1540.85 | 1533.4 | 1581.82 |
| **120** | **526.768** | 607.71 | 654.92 | 686.2 | 1495.66 | **2142.036** | 2363.37 | 2367.59 | 2381.2 | 2418.9 |
| **160** | **780.356** | 819.97 | 917.67 | 932.33 | 1912.08 | **3038.373** | 3176.17 | 3183.8 | 3197 | 3246.69 |
| **200** | **906.207** | 940.87 | 1067.49 | 1065.94 | 1905.7 | **3692.464** | 3755.21 | 3767.37 | 3778.3 | 3835.4 |
| **250** | **1187.904** | 1270.96 | 1490.65 | 1479.84 | 3054.8 | **4745.663** | 4988.94 | 5017.17 | 5007.6 | 5079.06 |
| **300** | 1481.412 | **1473.79** | 1765.49 | 1712.76 | 3309.14 | **5776.545** | 5832.69 | 5877.41 | 5862.5 | 5935.94 |
| **350** | **1681.50** | 1949.04 | 2010.74 | 1911.27 | 3638.19 | 6649.928 | **6607.52** | 6653.37 | 6632.4 | 6738.19 |
| **400** | **1933.379** | 1944.58 | 2421.98 | 2300.51 | 4347.64 | **7645.429** | 7738.56 | 7816.04 | 7760 | 7875.39 |
| **450** | 2298.365 | **2235.16** | 2840.79 | 2751.29 | 5350.35 | **8810.026** | 8845.9 | 8926.57 | 8876.3 | 9016.59 |
| **500** | **2475.335** | 2503.09 | 3174.39 | 3067.26 | 6023.74 | **9693.355** | 9902.64 | 9995.97 | 9921.8 | 10097.7 |

*Received 3/August/2021; Accepted 18/October/2021*

*ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.1.9

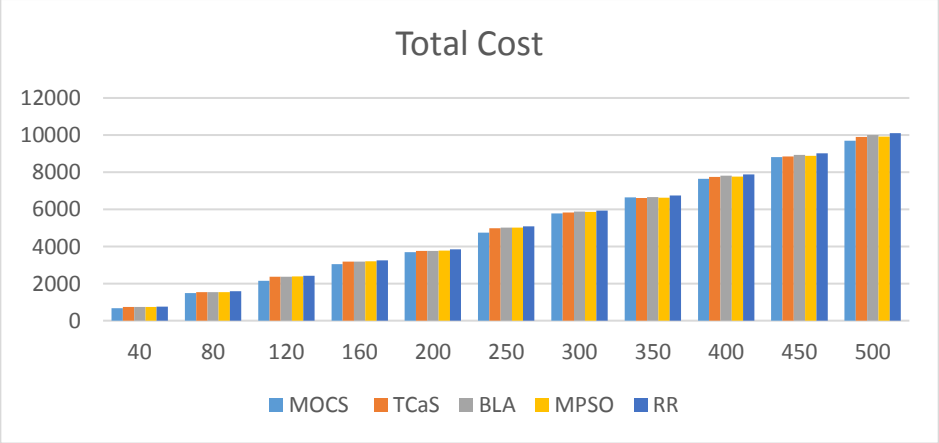


FIG. 1. TOTAL COST COMPARISON OF THE FIVE ALGORITHMS.

Total–cost for five strategies is compared in *Fig. 1*. Every dataset with a high average fitness showed that our suggested algorithm, MCSO, dominated the first place. Meanwhile, *Fig.* 2 compares the Makespan of the proposed model MCSO algorithm to the four others while the number of tasks changed, while *Fig.* 3 explained competition the time and makespan, demonstrating that our suggested MCSO algorithm outperforms the others. Based on the results, our suggested method, MCSO, 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. In complicated situations, MPSO proved to be more powerful than BLA. The RR algorithm produced poor results due to its simplistic methodology.

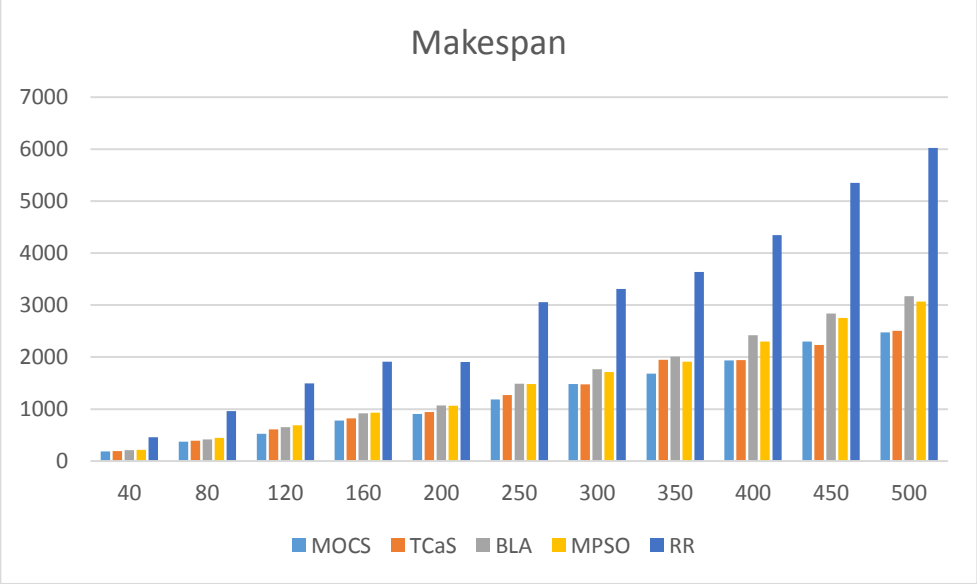


FIG. 2. MAKESPAN COMPARISON OF THE FIVE ALGORITHMS

*Received 3/August/2021; Accepted 18/October/2021*

*ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.1.9

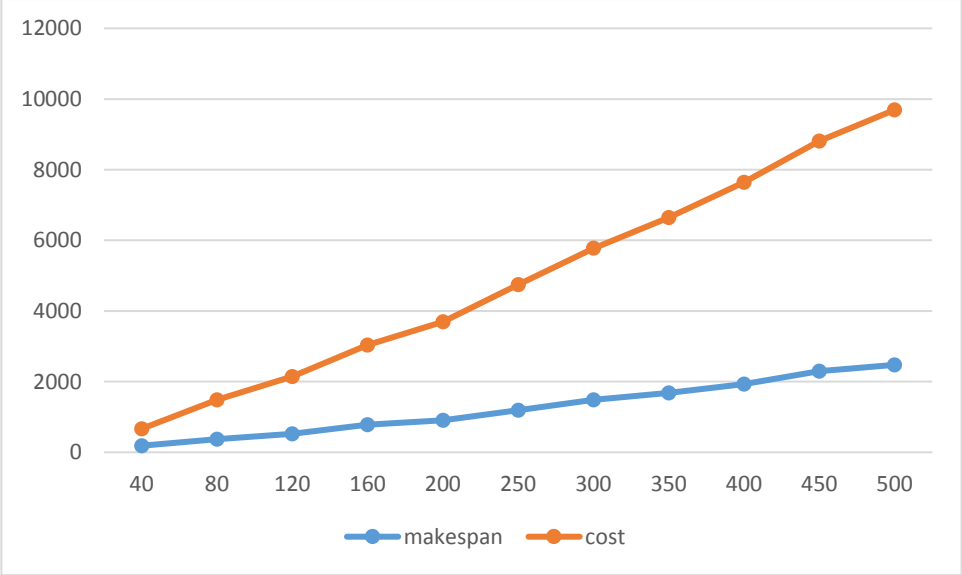


FIG. 3. TIME AND MAKESPAN COMPARISON OF THE MOCS ALGORITHMS

## VI. CONCLUSION

This paper provides a multiobjective cuckoo search optimization (MCSO) metaheuristic technique to solve the problem of task scheduling in a cloud computing context. Task scheduling in cloud computing is an NP-hard problem, which means that any deterministic solution will fail to produce the desired results. When compared to a single objective function, the multi-objective optimization strategy improves scheduling performance. The results suggest that our method was eminent to others. In terms of cost and time, it outperforms the existing BLA, MPSO, TCaS, and RR algorithms.

As a future work, we will examine offering more advanced ways to further optimize the balance between exploration and exploitation in the MCSO approach to obtain improved performance on convergence speed and accuracy in task scheduling. Meanwhile, we will investigate parallel implementations of our methodology in cloud environments to reduce the method's scheduling overhead in the presence of enormous workloads. We will also add some more advanced features to the suggested performance model and method for cloud computing task scheduling. For example, we'll aim to solve difficulties with Quality of Service (QoS), where some jobs are prioritized higher than others. There is also an intention to apply our method to more complicated task jobs, such as workflows, tasks that are not independent of one another, and cloud-based deep learning workloads.

## REFERENCES

[1] S. Srichandan, T. A. Kumar, and S. Bibhudatta, "Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm," *Futur. Comput. Informatics J.*, vol. 3, no. 2, pp. 210–230, 2018.

[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *2008 grid computing environments workshop*, 2008, pp. 1–10.

[3] Z.-G. Chen, K.-J. Du, Z.-H. Zhan, and J. Zhang, "Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, 2015, pp. 708–714.

[4] A. T. S. Al-Obaidi and S. A. Hussein, "Two improved cuckoo search algorithms for solving the flexible job-shop scheduling problem," *Int. J. Perceptive Cogn. Comput.*, vol. 2, no. 2, 2016.

[5] A. T. S. Al-Obaidi and H. S. Abdullah, "Camel herds algorithm: A new swarm intelligent algorithm to solve optimization problems," *Int. J. Perceptive Cogn. Comput.*, vol. 3, no. 1, 2017.

[6] A. Certa, G. Galante, T. Lupo, and G. Passannanti, "Determination of Pareto frontier in multi-objective maintenance optimization," *Reliab. Eng. Syst. Saf.*, vol. 96, no. 7, pp. 861–867, 2011.

Received 3/August/2021; Accepted 18/October/2021

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.1.9

[7] G. Chiandussi, M. Codegone, S. Ferrero, and F. E. Varesio, "Comparison of multi-objective optimization methodologies for engineering applications," *Comput. Math. with Appl.*, vol. 63, no. 5, pp. 912–942, 2012.
[8] Y. Kessaci, N. Melab, and E.-G. Talbi, "Multi-level and multi-objective survey on cloud scheduling," in *2014 IEEE International Parallel & Distributed Processing Symposium Workshops*, 2014, pp. 480–488.
[9] M. Kaur and S. Kadam, "A novel multi-objective bacteria foraging optimization algorithm (MOBFOA) for multi-objective scheduling," *Appl. Soft Comput.*, vol. 66, pp. 183–195, 2018.
[10] Q. Bai, "Analysis of particle swarm optimization algorithm," *Comput. Inf. Sci.*, vol. 3, no. 1, p. 180, 2010.
[11] G. Pan, Y. Xu, A. Ouyang, and G. Zheng, "An improved artificial chemical reaction optimization algorithm for job scheduling problem in grid computing environments," *J. Comput. Theor. Nanosci.*, vol. 12, no. 7, pp. 1300–1310, 2015.
[12] R. Salimi, H. Motameni, and H. Omranpour, "Task scheduling using NSGA II with fuzzy adaptive operators for computational grids," *J. Parallel Distrib. Comput.*, vol. 74, no. 5, pp. 2333–2350, 2014.
[13] T. P. Jacob and K. Pradeep, "A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization," *Wirel. Pers. Commun.*, vol. 109, no. 1, pp. 315–331, 2019.
[14] M. Abd Elaziz, S. Xiong, K. P. N. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowledge-Based Syst.*, vol. 169, pp. 39–52, 2019.
[15] V. M. A. Xavier and S. Annadurai, "Chaotic social spider algorithm for load balance aware task scheduling in cloud computing," *Cluster Comput.*, vol. 22, no. 1, pp. 287–297, 2019.
[16] R. K. Jena, "Task scheduling in cloud environment: A multi-objective ABC framework," *J. Inf. Optim. Sci.*, vol. 38, no. 1, pp. 1–19, 2017.
[17] M. Agarwal and G. M. S. Srivastava, "Genetic algorithm-enabled particle swarm optimization (PSOGA)-based task scheduling in cloud computing environment," *Int. J. Inf. Technol. Decis. Mak.*, vol. 17, no. 04, pp. 1237–1267, 2018.
[18] B. A. Al-Maytami, P. Fan, A. Hussain, T. Baker, and P. Liatsis, "A task scheduling algorithm with improved makespan based on prediction of tasks computation time algorithm for cloud computing," *IEEE Access*, vol. 7, pp. 160916–160926, 2019.
[19] H. Xu, B. Yang, W. Qi, and E. Ahene, "A multi-objective optimization approach to workflow scheduling in clouds considering fault recovery," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 3, pp. 976–995, 2016.
[20] A. Alzaqebah, R. Al-Sayyed, and R. Masadeh, "Task scheduling based on modified grey wolf optimizer in cloud computing environment," in *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, 2019, pp. 1–6.
[21] J. Zhou, T. Wang, P. Cong, P. Lu, T. Wei, and M. Chen, "Cost and makespan-aware workflow scheduling in hybrid clouds," *J. Syst. Archit.*, vol. 100, p. 101631, 2019.
[22] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterp. Inf. Syst.*, vol. 12, no. 4, pp. 373–397, 2018.
[23] H. T. T. Binh, T. T. Anh, D. B. Son, P. A. Duc, and B. M. Nguyen, "An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment," in *Proceedings of the Ninth International Symposium on Information and Communication Technology*, 2018, pp. 397–404.
[24] A. M. S. Kumar and M. Venkatesan, "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment," *Wirel. Pers. Commun.*, 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," *J. Water Resour. Plan. Manag.*, 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," *Wirel. Pers. Commun.*, 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 Trans. Cloud Comput.*, vol. 7, no. 2, pp. 537–552, 2016.
[28] B. M. Nguyen, H. Thi Thanh Binh, and B. Do Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment," *Appl. Sci.*, vol. 9, no. 9, p. 1730, 2019.
[29] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *2009 World congress on nature & biologically inspired computing (NaBIC)*, 2009, pp. 210–214.
[30] S. Abdi, S. A. Motamedi, and S. Sharifian, "Task scheduling using modified PSO algorithm in cloud computing environment," in *International conference on machine learning, electrical and mechanical engineering*, 2014, vol. 4, no. 1, pp. 8–12.
[31] I. S. Rajput and D. Gupta, "A priority based round robin CPU scheduling algorithm for real time systems," *Int. J. Innov. Eng. Technol.*, vol. 1, no. 3, pp. 1–11, 2012.

*Received 3/August/2021; Accepted 18/October/2021*
 *ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*