Efficient Task Scheduling in Cloud Systems with Adaptive Discrete Chimp Algorithm

Emrullah Gunduzalp, Gungor Yildirim and Yetkin Tatar

Abstract— **Successful task scheduling is one of the priority actions to increase energy efficiency, commercial earnings, and customer satisfaction in cloud computing. On the other hand, since task scheduling processes are NP-hard problems, it is difficult to talk about an absolute solution, especially in scenarios with large task numbers. For this reason, metaheuristic algorithms are frequently used in solving these problems. This study focuses on the metaheuristic-based solution of optimization of makespan, which is one of the important scheduling problems of cloud computing. The adapted Chimp Optimization Algorithm, with enhanced exploration and exploitation phases, is proposed for the first time to solve these problems. The success of the proposed method has been tested for different simulation scenarios. According to the simulation results, the proposed method achieved a makespan improvement of approximately 30% compared to the standard task scheduling algorithms.**

Index Terms— **Chimp algorithm, Cloud computing, Makespan, Metaheuristic, Optimization.**

I. INTRODUCTION

CLOUD COMPUTING is technology that allows the dynamic delivery of flexible, scalable, and distributed computing resources to end-users [1]. Many reasons such as the development of the Internet infrastructure, the use of the Internet of Things (IoT) technology, the increase in the need for big data technologies, and the developments in artificial intelligence have led to the widespread use of cloud computing. Cloud computing enables users to access various services and resources (CPU, RAM, storage) wherever there is internet access. Thanks to the pay-as-you-go system, both software and hardware resources become more economical, while installation and maintenance costs are significantly reduced for customer companies [2]. Cloud service providers must provide resources and services to their customers without violating the Service Level Agreement (SLA) and guaranteeing Quality of Service (QoS). Therefore, cloud performance is very important for both service providers and users. One of the main issues affecting the performance of cloud systems is task scheduling. Task scheduling is one of the important problems in cloud computing. Especially, inefficient task scheduling could cause loss of performance and revenue, and SLA violation. Efficient scheduling algorithms can optimize important measurements such as makespan, traffic volume, computational time, communication cost, system efficiency and utilization [3].

Virtual machines (VMs), one of the basic mechanisms of cloud technologies, contribute to the efficient use of infrastructure resources. The task scheduling process in this study can be briefly described as follows: Distributing tasks of different sizes reaching the cloud system to the most suitable VMs in a way that provides the shortest response time. Inappropriate scheduling can reveal underloaded (underutilized) or overloaded (overutilized) of resources, referred to as resource dilemma. These situations ultimately lead to wasted cloud resources or reduced service performance [4]. In solving NP-hard problems, using metaheuristic methods instead of standard deterministic algorithms may yield more successful results [1]. In this study, the task scheduling process in cloud systems is studied with an adapted version of Chimp Optimization Algorithm (ChOA), which is a metaheuristic algorithm in use. The makespan problem was taken as a basis in the scheduling process. To the best of the authors' knowledge, the proposed adaptive version of the ChOA and its application on this type of cloud system problem has never been studied before in the literature. Simulations were carried out by integrating ChOA into the CloudSim 3.0.3 simulator. First of all, the Adaptive Chimpanzee Optimization Algorithm (AChOA) has been proposed, which makes the exploration and exploitation stages of ChOA more adaptive and uses different mathematical functions for this purpose. In AChOA, exploration and exploitation mechanisms, apart from the standard method, Sigmoid Decreasing Weight (SDW), Oscillating Weight (OscW) and V-shaped Family (VSW) functions have been included in the calculation process. Thus, by using different functions, the efficiency and performance of the optimization process are comparatively observable. Finally, the space shared task scheduling approach of both AChOA and CloudSim, which use different functions, are tested for different experimental scenarios and the results are discussed. The original aspects of this study can be briefly summarized as follows:

EMRULLAH GUNDUZALP, is with Department of Computer Engineering University of Firat University, Elazig, Turkey, (e-mail: emrullahg@dsi.gov.tr).
https://orcid.org/0000-0001-6418-5663

GUNGOR YILDIRIM, is with Department of Computer Engineering University of Firat University, Elazig, Turkey,,(e-mail: gungor.yildirim@firat.edu.tr).
https://orcid.org/0000-0002-4096-4838

YETKIN TATAR is with Department of Computer Engineering University of Firat University, Elazig, Turkey,,(e-mail: ytatar@firat.edu.tr).
https://orcid.org/0000-0002-7181-0014

Manuscript received September 1, 2021; accepted July 29, 2022.
DOI: 10.17694/bajece.989467

- The study uses the metaheuristic Chimp algorithm for the first time for task scheduling problems in cloud systems.
- Unlike other swarm-based metaheuristic algorithms, the proposed method adaptively manages exploration and exploitation processes.
- Compared to the standard scheduling method, the proposed method achieves an improvement of approximately 30% in makespan.
- The adaptive method used can be easily adapted to other optimization problems that uses the Chimp algorithm.

The adaptive method used can be easily adapted to other problems using the Chimp algorithm. The continuation of this article is organized as follows. In section II, the task scheduling problem in cloud systems is introduced and literature studies on this subject are explained. In Section III, the methodology of the method applied in this study is presented. Experiments and analyzes are shared in section IV, and conclusions are given in section V.

II. Task Scheduling and Literature Summary in Cloud Computing

Task scheduling is a process that affects the performance and efficiency of cloud systems. In short, task scheduling can be expressed as the optimal assignment of n tasks $\{T_1, T_2, \ldots, T_n\}$ to m machines$\{M_1, M_2, .., M_m\}$, taking into account one or more predefined optimization targets [4]. With virtualization techniques, which have provided significant advantages in recent years, physical servers used in cloud systems can be divided into more than one virtual machine (VM), and each virtual machine can be used to allocate different tasks. Cloud service providers (CSPs) may have multiple observer and control infrastructure services. One of the most important of these is the broker services that optimally distribute the incoming tasks to the resources in the system according to their types and characteristics. While doing this, they use algorithms that take into account both the task and the resource characteristics offered.

Task scheduling in cloud computing generally includes three main operations [6]. These are Strategy Phase, Planning Phase, and Deployment Phase. In Strategy Phase, all shared resources in the data center and their properties are made discoverable and questionable. In Planning Phase, a suitable resource is determined according to the task requirements, while in Deployment Phase, the selected resources are allocated to the relevant tasks. It is a complex process to carry out the planning process in a heterogeneous and dynamic environment such as the cloud environment. It is quite difficult to find the optimum planning method in this process, which is carried out with different optimization objectives such as cost, energy consumption, makespan, and execution variability. Target strategies are generally grouped under four headings [7]. The scheduling strategies are dynamism, target architecture and scheduling algorithms.

Task scheduling can be single-objective or multi-objective. While optimizing the scheduling process, one of the goals, such as makespan, computational cost, or several goals contradicting with each other can be taken as a basis. At the same time, the scheduling process can be done statically or dynamically. Static algorithms can produce successful and fast results in small-scale cloud systems where prior knowledge of incoming tasks and available resources is not required, the workload does not change frequently. However, cloud systems are by nature dynamic systems with a lot of variation in workload. Therefore, dynamic algorithms give more successful results. [4,7]. The target system architecture is another issue that affects the scheduling strategy to be implemented. Suitable solution algorithms are determined during the planning process. The algorithms used here are classified as heuristics, metaheuristics, and hybrid algorithms. Heuristic algorithms are often preferred for static scheduling. These algorithms are fast, but they are insufficient for cloud systems with a large-scale dynamic environment. Metaheuristic strategies are effective methods for solving NP-hard optimization problems with high efficiency. These algorithms can be expressed as Heuristic + Randomization. Another approach to solving the task scheduling problem is the use of hybrid methods in which two or more algorithms are combined. [4]. In Fig.1, examples of algorithms used in the literature according to this classification are given.

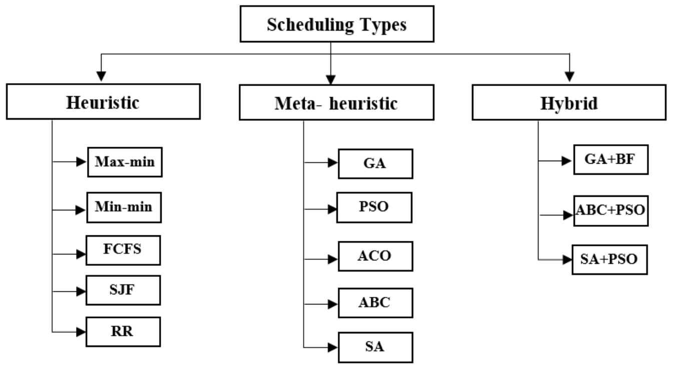


Fig.1. Various scheduling algorithms [6,7]

Many problems encountered in cloud computing attract the attention of researchers. One of these problems is the task scheduling process. In [8], the authors use the minimum completion time (MCT) and longest job to fastest processor (LJFP) to initialize the PSO. The goal is to minimize the makespan, total energy consumption and execution time, and degree of imbalance. In [9], the authors used the gray wolf optimization (GWO) technique to solve the task scheduling problem. The authors aimed to minimize the makespan. In [10], the authors proposed a whale optimization algorithm (WOA)-based method that aims to improve the performance of task scheduling with a multi-objective optimization model. They claimed that they improved their ability to search for optimum solutions with the approach they named IWC. In another study [11], the authors proposed Electromagnetism Metaheuristic Algorithm (EMA) in order to improve QoS in the cloud. They achieved this by scheduling tasks on virtual machines (VMs) to optimize completion time. In [12], the authors analyzed it with a time-shared and space-shared genetic algorithm. The study states that the method used outperforms competitive scheduling methods in terms of completion time and cost. In [13], the authors introduced an ACO-based load balancing algorithm for task scheduling. A hybrid approach using PSO and ACO algorithms for task scheduling was proposed in [14]. Another hybrid approach is introduced in [15]. In this approach, the genetic algorithm uses its global search capability to minimize the task execution time and then transforms the obtained results into the initial pheromone of ACO to achieve more successful optimization. In [16], the authors proposed a hyper-heuristic scheduling algorithm using a framework including GA, PSO, and ACO in order to optimize makespan.

III. Problem Definition and Methodology

Tasks submitted by users are put into task queues before they are assigned to the respective virtual machines. Tasks waiting in the queue are sent to the task planner by the VMM (Virtual Machine Manager). The task planner determines the resources that will execute the tasks and makes the assignments to the relevant VMs in a way as to use the resources efficiently [9]. In this study, this process was performed with AChOA. Task assignment is usually performed using the tabulation technique. In this technique, first of all, it is determined which task will be executed by which virtual machines. For example, mapping 10 tasks, belonging to a specific application and not dependent on each other's output, to m=5 virtual machines can be shown in Fig.2.

VM$_5$	VM$_1$	VM$_2$	VM$_3$	VM$_1$	VM$_5$	VM$_5$	VM$_1$	VM$_2$	VM$_2$
T$_1$	T$_2$	T$_3$	T$_4$	T$_5$	T$_6$	T$_7$	T$_8$	T$_9$	T$_{10}$

Fig.2. An example task scheduling representation

The task group, T=$\{T_1, T_3, .., T_n\}$, is a set of independent tasks containing million instructions (MI). Each task in T is limited to the specified number of commands. The V cluster contains VMs, and each VM has a metric that shows how many million instructions per second (MIPS) it can process. The MIPS value of the VMs is also within the specified limits. Assume that the sizes of the tasks are as in Fig.3 after they have been assigned to the VMs as in the example in Fig.2. In this case, the makespan value for this task set will be the total time that VM$_5$ will spend executing T$_1$, T$_6$, and T$_7$.

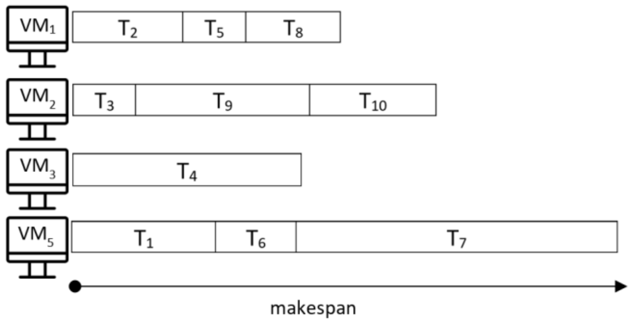


Fig.3. Tasks assigned to VMs according to an example scenario

In the representation of these types of assignment operations, it is usually used a matrix model that shows which task is assigned to which VM. This matrix is as follows.

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}, \text{here} \qquad (1)$$

$$\begin{cases} x_{ij} = 1, & if\ T_i\ is\ assigned\ to\ VM_j \\ x_{ij} = 0, & if\ T_i\ is\ not\ assigned\ to\ VM_j \end{cases}$$

As a result of the above-mentioned definitions, the estimated execution time for the ith task in jth VM is calculated by Eq.2. Since the tasks are independent, each task can only be assigned to one VM, and this is expressed as $\sum_{j=0}^{m} x_{ij} = 1,\ (1 \leq i \leq n)$. Thus, the execution time (VETj) of jth VM is calculated as in Eq.3.

$$ECT_{ij} = \frac{T_i}{VM_j} \qquad (2)$$

$$VET_j = \sum_{j=0}^{n} x_{ij} * ECT_{ij}, \qquad 1 \leq j\ m \qquad (3)$$

The makespan (MS) in a task group is equal to the maximum execution time, expressed by Eq 4. The objective function used is to minimize the MS for all candidate solutions.

$$MS = Max\left\{VET_j\right\},\ 1 \leq j \leq m \qquad (4)$$

A. *Chimp Optimization Algorithm*

Metaheuristic algorithms are among the techniques that are frequently used in solving optimization problems in the literature. Population-based algorithms using swarm intelligence constitute an important part of metaheuristic algorithms. In this section, one of the current algorithms, Chimp Optimization Algorithm-ChOA will be explained and the application of this algorithm to task scheduling in cloud systems will be shown. In the study in [5], the authors developed the ChOA inspired by the hunting strategies of chimps. According to this algorithm, there are four different chimps in a colony. Those are the attacker, chaser, barrier, driver. They all have different abilities and they use these abilities while hunting. In this algorithm, the attacker is the candidate that holds the best result. The hunting mechanism in ChOA consists of two phases. The first is the exploration phase, in which driving and chasing the prey are performed. This is expressed by Eq. 5-6.

$$d = |\vec{c} \circ \vec{X}_{prey}(t) - \vec{m} \circ \vec{X}_{chimp}(t) \qquad (5)$$

$$\vec{X}_{chimp}(t+1) = \vec{X}_{prey}(t) - \vec{a}.d \qquad (6)$$

In this equation, t represents the current iteration, while c, m, and a are the coefficient vectors calculated by Eq. 7-9. $\vec{X}_{prey}$ and $\vec{X}_{chimp}$ are the position vectors of prey and predator, respectively. ° represents the Hadamard product.

$$\vec{a} = 2.f.\vec{r_1} - f \qquad (7)$$

$$\vec{c} = 2.\vec{r_2} \qquad (8)$$

$$\vec{m}_{(t+1)} = \begin{cases} 1, & if\ \vec{m}_t = 0 \\ \dfrac{1}{mod(\vec{m}_t, 1)}, & if\ \vec{m}_t \neq 0 \end{cases} \qquad (9)$$

Here, the f value is reduced from 2.5 to 0 depending on the current iteration value. r_1 and r_2 are random uniform vectors. The *m* value is a chaotic vector representing the effect of chimps' intuitive motivation in the hunting process. In this study, this chaotic value was calculated by Gauss/Mouse map [23] method as in Eq. 9. The elements of the vector $\vec{c}$ change randomly in the interval [0,2]. It also improves the ChOA's stochastic behaviour and reduce the likelihood of being caught at the local minimum.

The second stage is the exploitation (attack stage). In this stage, the attacker, chaser, barrier, and driver have information about the prey location. Thus, the four best solutions obtained so far are retained, and the other chimpanzees update their positions to these four best chimp positions, as in Eqs. 10-12.

$$\begin{aligned} d_{attacker} &= |\vec{c}_1 \circ \vec{X}_{attacker} - \vec{m}_1 \circ \vec{X}_{candidate}|, \\ d_{barrier} &= |\vec{c}_2 \circ \vec{X}_{barrier} - \vec{m}_2 \circ \vec{X}_{candidate}|, \\ d_{chaser} &= |\vec{c}_3 \circ \vec{X}_{chaser} - \vec{m}_3 \circ \vec{X}_{candidate}|, \\ d_{driver} &= |\vec{c}_4 \circ \vec{X}_{driver} - \vec{m}_4 \circ \vec{X}_{candidate}| \end{aligned} \tag{10}$$

$$\begin{aligned} x_1 &= x_{attacker} - a_1(d_{attacker}), \\ x_2 &= x_{barrier} - a_2(d_{barrier}), \\ x_3 &= x_{chaser} - a_3(d_{chaser}), \\ x_4 &= x_{driver} - a_4(d_{driver}) \end{aligned} \tag{11}$$

$$x(t+1) = \frac{x_1 + x_2 + x_3 + x_4}{4} \tag{12}$$

It is assumed that the chimps have a 50% probability of choosing between the normal update or the chaotic update mechanism. This is modeled is by Eq.13. In this equation, μ is a random variable between 0 and 1.

$$\vec{X}_{chimp}(t+1) = \begin{cases} \vec{X}_{prey}(t) - \vec{a}.d, & if\ \mu < 0.5 \\ m(t) \quad , & if\ \mu > 0.5 \end{cases} \tag{13}$$

On the other hand, there is an approach in the literature where the next positions of the candidate solutions are calculated with the weighted values of x_1, x_2, x_3 *and* x_4 [24]. However, the method proposed in this study uses traditional Eq.11 and 12.

B. *Adapted Chimp Optimization Algorithm (AChOA)*

In the ChOA algorithm, |a|<1 forces chimpanzees to attack the prey (exploitation), while |a|>1 causes chimpanzees to scatter in search of better prey (exploration). This parameter depends on two important sub-parameter changes. The first of these is the random vector r_1 and can be obtained as either uniform or chaotic. The second subparameter is f. In classical ChOA, this value is calculated to be reduced from 2.5 to 0. The adaptability of this sub-parameter will also make the exploration and exploitation mechanisms adaptive. Adaptive metaheuristic approaches are known to provide performance improvement in solving different problems in the literature [17,22]. This study has implemented the ChOA and its adaptive version to the task scheduling problem in cloud systems. As far as is known, this is the first time in the literature. In AChOA, three different mathematical methods given below are suggested for calculating the f coefficient.

Sigmoid Decreasing Weight (SDW): SDW uses a sigmoid function [17, 22]. This function, given in Eq.14, generates a value by using the upper(U), lower(L) bounds, and the current iteration (t). In this equation, T_{max} is the maximum iteration, and u is calculated with $u = 10^{(\log(T_{max})-2)}$.

$$f(t) = U + \frac{U - L}{(1 + e^{u(0.5.T_{max} - t)})} \tag{14}$$

Oscillating weight (OscW): In this method, a waveform is used for discovery and exploitation processes [18, 22]. The main function of this waveform is given in Eq.15. In this equation, S depends on S_1 and they are calculated by $S_1 = \frac{3T_{max}}{4}$ and $S = \frac{2S_1}{3+2k}$. The k value is a predefined constant.

$$f(t) = \frac{U + L}{2} + \frac{U - L}{2} Cos(\frac{2\pi t}{S}) \tag{15}$$

V-shape Family (VSW): This function, which is frequently used in binary search PSO algorithms, is not compelling in the displacement of solutions [19, 22]. The general expression of VSW is as in Eq.16.

$$f(t) = \left|\frac{\pi}{2} \mathrm{Tan}^{-1}(\frac{\pi}{2} t)\right| \tag{16}$$

C. *Discrete AChOA and Application to the Task Scheduling Problem*

The fact that virtual machine representation formats are generally integer encoded allows task scheduling algorithms to be discrete. This study also uses discrete value representation for the task scheduling optimization algorithm. For this reason, the discrete form of AChOA was developed. The discrete optimization of AChOA, which will assign tasks to *m* VMs with certain MIPS values that have been created before, consists of five steps.

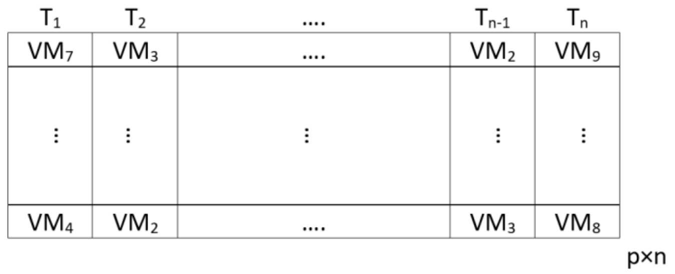


Fig.4. A sample initial population

<u>Step 1</u>*: Initializing the population:* In the first step, *n* tasks are generated with random MI values within the maximum and minimum limits. According to the number of tasks (*n*), the initial positions of all chimpanzees are determined. In this step, random candidate solutions are generated for the population of size *p* as in Figure 4.

Step 2: Creation of Expected Time to Compute Matrix (ETC): In a candidate solution as in Figure 4, the ETC matrix is created by checking the assignment of the tasks to the VM as in Eq.2.

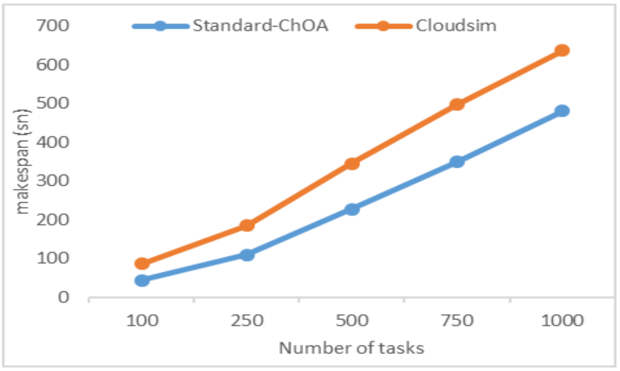
Step 3: Calculation of the maximum makespan: In a population where an ETC matrix is created as in Eq. 2, the maximum makespans are calculated for each candidate solution, as in Eqs. 1-4. The best value found is compared with the scores of the attacker, chaser, barrier, and driver chimps. Then the information of the chimp with the best position and the best score are updated accordingly.

Step 4: Updating the search location: The coefficients are recalculated according to Eqs.7-9. The search position of the entire population is updated by Eqs.10-12. As a result, a new candidate solution is created, replacing the tasks initially randomly assigned to the VMs.

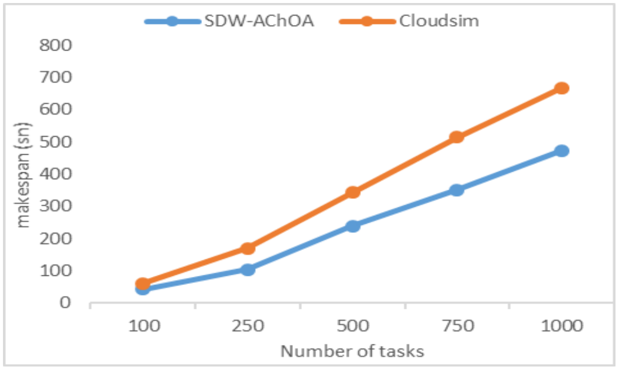
Step 5: Finalizing the iteration: Steps 2-4 are repeated until the maximum number of iterations is reached. As a result of iterations, the attacker score gives the optimum makespan and VM-task match according to AChOA. The pseudocode of the task scheduling process with AChOA is given in Fig. 5.

```
Task scheduling with AChOA
  Set initial positions of attacker, barrier, chaser and driver chimps
  Generate random tasks and virtual machines based on MI and MIPS boundaries
  Start chimp population $x_i$ (i=1,2, …,n)
  Set f, m, a and c values
  $X_{attacker}$= best search agent, $P_{attacker}$= best position
  $X_{barrier}$= second best search agent, $P_{barrier}$= second best position
  $X_{chaser}$= third best search agent, $P_{chaser}$= third best position
  $X_{driver}$= fourth best search agent, $P_{driver}$= fourth best position
    while (t < maximum number of iterations)
        for each chimp:
            Check boundary conditions
            Calculate the fit (max makespan) value based on the objective function
            if(fitness value < $X_{attacker}$)
                $X_{attacker}$ = fitness value
                $P_{attacker}$ = chimp position
            if($X_{barrier}$ < fitness value > $X_{attacker}$)
                $X_{barrier}$ = fitness value
                $P_{barrier}$ = chimp position
            if($X_{chaser}$ < fitness value > $X_{attacker}$, $X_{barrier}$)
                $X_{chaser}$ = fitness value
                $P_{chaser}$ = chimp position
            if($X_{driver}$ < fitness value > $X_{attacker}$, $X_{barrier}$, $X_{chaser}$)
                $X_{driver}$ = fitness value
                $P_{driver}$ = chimp position
        end for
        update f value
        for each chimp:
            calculate a, c, m values (Eqs. 7-9)
            update chimp position (Eqs. 10-14)
      end for
      t=t+1
    end while
  return $X_{attacker}$
```

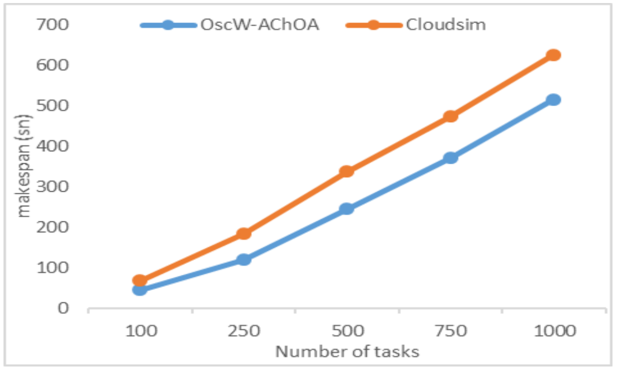
Fig.5. Task scheduling pseudocode with AChOA



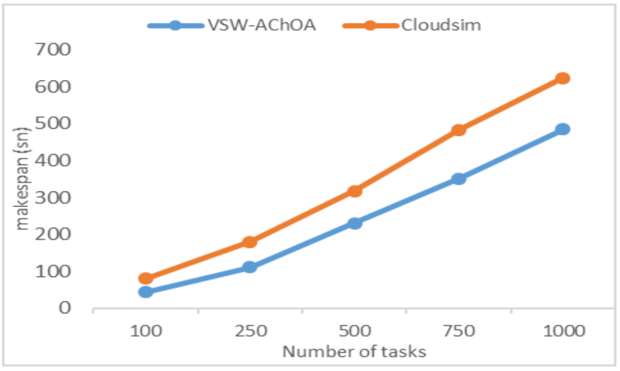
a) Standard-ChOA and Cloudsim makespan results



b) SDW-AChOA and Cloudsim makespan results



c) OscW-AChOA and Cloudsim makespan results



d) VSW-AChOA and Cloudsim makespan results

Figure 6. Comparison of time to completion (makespan) performance for all methods

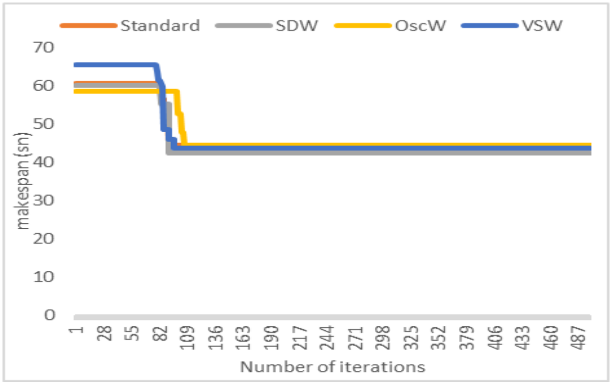
IV. Experiments

CloudSim simulator was used to prove the accuracy of AChOA proposed for task scheduling. To be compatible with this simulator, ChOA is written in the JAVA programming language. CloudSim is one of the most popular simulators used by researchers and engineers doing research and development for cloud-related problems [20]. CloudSim simulator enables the modelling of main cloud system entities (data centers, mainframes, VMs, and cloudlets, etc.). Besides, it allows examining scheduling approaches in the created cloud environment [21]. The parameters used for the simulations are as in Table I.

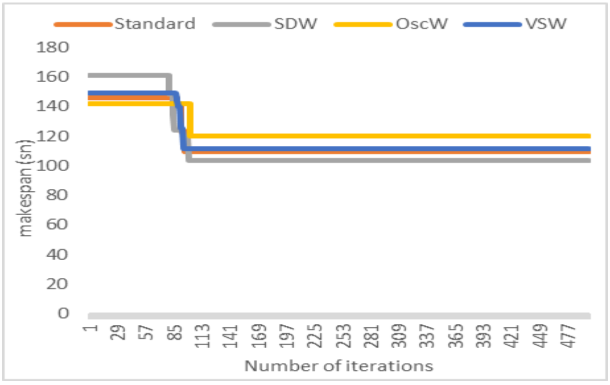
TABLE I
SIMULATION PARAMETERS

Parameter	Value
Population size	50
Number of iterations	500
Number of tasks	100-1000
Number of VMs	20
Task command length (MI)	5000-20000
VM processing capacity (MIPS)	1000-5000

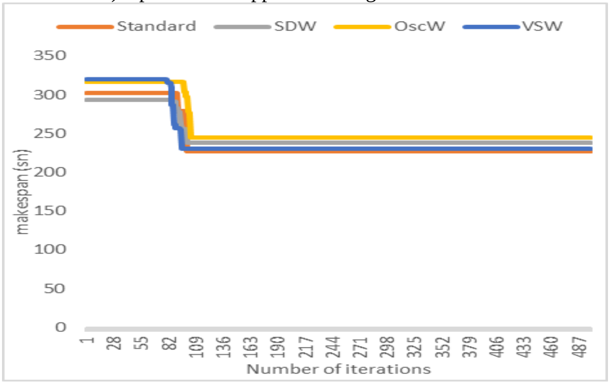
Task scheduling can be made on a real cloud system or a user-defined virtual system on the cloud. In the simulations, custom task scheduling scenarios, which are carried out on a user-defined virtual system on the cloud, have been considered. In the experiments, simulations of AChOA using both standard ChOA and different adaptive functions were run separately. Obtained results are also discussed in comparison with CloudSim default task scheduling results. To observe the performances in different scenarios, each experiment carried out with 100, 250, 500, 750, and 1000 tasks were run 20 times, and statistical results were obtained. In practice, customers generally purchase cloud services on a pay-as-you-go model. Although nowadays there are more advanced server and virtual machine specifications (>100000MIPS, >2GB memory), economical specifications of the hardware used have been chosen according to realistic customer scenario suitability. For this reason, MIPS values are generally close to the standard Intel Pentium specifications, and the task sizes are chosen according to the traditional image processing and scientific algorithms (>5000) used in the literature. First, two main physical servers for VMs were created in the data center, each with 16 GB ram, 10 TB storage, 1 GB/s bandwidth, and time-sharing VM scheduling.



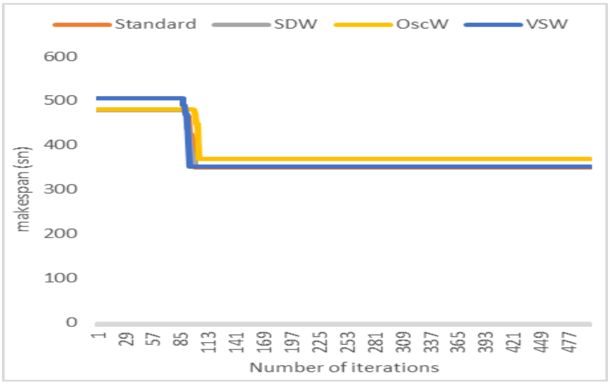
a) Optimization approach changes for 100 tasks



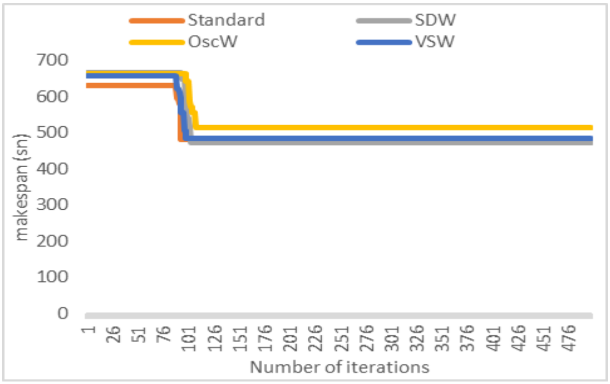
b) Optimization approach changes for 250 tasks



c) Optimization approach changes for 500 tasks



d) Optimization approach changes for 750 tasks



e) Optimization approach changes for 1000 tasks

Figure 7. Optimization approach changes for different number of tasks

All VMs are equally shared on these physical servers. The first of these computers has a 4-core CPU and the second has a dual-core X86 architecture CPU. The processing capacity of each processor core is 10000 MIPS. The computers have Linux operating system and Xen VMM. VMs have 512 MB ram, 10 GB storage, 10 MB/s bandwidth, and time-sharing task scheduling configuration. The processing capacity of the VMs varies between 1000 and 5000 MIPS. The instruction length of the tasks varies between 5000 and 20000 MI. On the CloudSim side, the standard task scheduling method *CloudletSchedulerSpaceShared* is used.

In the experimental results, firstly, the makespan performances of all methods were examined. In Fig.6, the simulation results of each method and CloudSim are given.

Since the task MI values are randomly determined in each simulation, the same task types in that simulation are also given to CloudSim for accurate evaluation. Therefore, comparisons are given separately in the four charts. The general evaluation between the methods is presented with the statistical data in Table II.

While SDW found the minimum makespan in three of the experiments performed with five different task numbers, the Standard method caught the best makespan in the other two. In all experiments performed, it was seen that all variations of ChOA achieved better results than CloudSim's standard space shared task scheduling process. Considering all the task numbers in this experiment, SDW-AChOA 35.5%, standard ChOA 32.2%, OscW-AChOA 27.2%, and VSW-AChOA 32% improved according to CloudSim. According to these average values, it is seen that the SDW method is better in performance. Another result obtained from the experiments is that as the number of tasks increases, the percentage of success decreases according to CloudletSchedulerSpaceShared. For example, while the improvement rate compared to CloudSim was 48.5% in the experiment with 100 tasks, which is performed by the Standard method, this rate decreased to 24.5% in the experiment with 1000 tasks. As the number of tasks increased, the most significant decrease was 50.6% in VSW-AChOA compared to CloudSim. While the improvement rate was 45.44% in the 100-tasks experiment performed with VSW-AChOA, this value decreased to 22.43% in the 1000-tasks simulation. However, while the decrease in the amount of improvement due to the increase in the number of tasks in OscW-AChOA was 49.18%, this rate was 49.46% in the Standard method. When this example was evaluated on an experimental basis, the consistent method was seen as SDW-AChOA.

To discuss the performance of all methods relative to each other in a more general perspective, it is necessary to evaluate the statistical results obtained from 20 independent experiments and given in Table II. Considering all the experimental results, it is seen that SDW -AChOA is more successful in reaching the minimum value for 100 and 250 tasks. However, this performance decreased as the number of tasks increased, even worse than other results. It has been seen that Standard ChOA is more successful at high task numbers. Another method that performs well for high tasks is VSW-AChOA. In fact, based on the mean value criterion, VSW-AChOA can be seen as relatively more successful than standard ChOA. As a result, SDW-AChOA performed better on small task numbers, while Standard and VSW-AChOA were successful in experiments involving a high number of tasks. Although all methods were more successful than standard scheduling methods, OscW had the lowest performance among AChOA variations.

A large number of candidate variables (tasks) also affected the approach to the best solution in the optimization process. The convergence to the best in all methods generally occurred after the 70th iteration, and the best was reached after about 50 iterations. Fig.7 shows the Standard, SDW, OscW, and VSW approximation results in the experiments.

TABLE II
STATISTICAL RESULTS OF EXPERIMENTS FOR DIFFERENT TASK NUMBERS

Method		100 tasks	250 tasks	500 tasks	750 tasks	1000 tasks
Standard-ChOA	**Minimum**	44,4	109,5	227,4	349,3	480,1
	Maximum	74,1	177,7	346,8	511,8	680,4
	Mean	51,2	126,4	261,5	399,4	538,8
	Standard Deviation	2,3	4,2	5,5	8,3	9,0
	Median	47,8	119,3	249,2	381,9	515,6
SDW-AChOA	**Minimum**	42,5	103,7	238,2	350,8	471,9
	Maximum	71,8	172,4	349,7	530,1	705,8
	Mean	50,9	128,1	262,1	397,8	534,9
	Standard Deviation	1,8	4,0	4,9	10,1	12,0
	Median	48,5	121,2	247,3	377,8	507,6
OscW-AChOA	**Minimum**	44,4	120,0	245,4	370,0	515,0
	Maximum	73,6	178,8	347,9	518,5	683,5
	Mean	52,5	136,1	280,1	420,7	571,8
	Standard Deviation	1,6	4,4	7,2	8,0	13,5
	Median	49,0	131,6	268,3	406,6	554,8
VSW-AChOA	**Minimum**	43,6	111,9	230,9	351,1	483,5
	Maximum	72,7	181,4	353,8	520,1	695,2
	Mean	50,8	127,3	263,5	393,8	538,8
	Standard Deviation	2,5	4,3	5,6	9,6	9,2
	Median	47,8	122,2	248,7	375,4	514,7

V. CONCLUSION

This study focused on task scheduling optimization, which is one of the most important problems in cloud computing. The solution to this problem was realized with AChOA, which is an improved adapted version of ChOA that is a new metaheuristic algorithm. AChOA uses functions that could make the exploration and exploitation mechanisms of the standard ChOA more flexible. Thus, this adapted metaheuristic algorithm can run in three different states: SDW-AChOA, OscW-AChOA, VSW-AChOA. The success of all methods was also proven in CloudSim, a well-known and successful cloud simulator in the literature. All metaheuristic methods achieved about 30% improvement over the standard scheduling method. Although the improvement decreased with the increase in the number of tasks, this rate was over 20%. In comparisons among themselves, it was observed that the results of SDW-AChOA in low task numbers and standard ChOA and VSW-AChOA in scheduling with high task numbers were more successful. Cloud systems will continue to be a research area where different problems will arise for a long time. For this reason, the authors will focus on the solutions of different optimization methods in cloud systems in their future studies.

REFERENCES

[1] Strumberger, I., Tuba, E., Bacanin, N., & Tuba, M. (2019, June). Dynamic tree growth algorithm for load scheduling in cloud environments. In 2019 IEEE Congress on Evolutionary Computation (CEC) (pp. 65-72). IEEE.

[2] Avram, M. G. (2014). Advantages and challenges of adopting cloud computing from an enterprise perspective. Procedia Technology, 12, 529-534.

[3] Abdullahi, M., & Ngadi, M. A. (2016). Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Generation Computer Systems, 56, 640-650.

[4] Houssein, E. H., Gad, A. G., Wazery, Y. M., & Suganthan, P. N. (2021). Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. Swarm and Evolutionary Computation, 62, 100841.

[5] Khishe, M., & Mosavi, M. R. (2020). Chimp optimization algorithm. Expert systems with applications, 149, 113338, https://doi.org/10.1016/j.eswa.2020.113338

[6] Pradhan, A., Bisoy, S. K., & Das, A. (2021). A survey on pso based meta-heuristic scheduling mechanism in cloud computing environment. Journal of King Saud University-Computer and Information Sciences.

[7] Saurav, S. K., & Benedict, S. (2021, January). A Taxonomy and Survey on Energy-Aware Scientific Workflows Scheduling in Large-Scale Heterogeneous Architecture. In 2021 6th International Conference on Inventive Computation Technologies (ICICT) (pp. 820-826). IEEE.

[8] Alsaidy, S. A., Abbood, A. D., & Sahib, M. A. (2020). Heuristic initialization of PSO task scheduling algorithm in cloud computing. Journal of King Saud University-Computer and Information Sciences.

[9] Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I., Tuba, M., & Zivkovic, M. (2019, November). Task scheduling in cloud computing environment by grey wolf optimizer. In 2019 27th Telecommunications Forum (TELFOR) (pp. 1-4). IEEE.

[10] [Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y., & Murphy, J. (2020). A woa-based optimization approach for task scheduling in cloud computing systems. IEEE Systems Journal, 14(3), 3117-3128.

[11] Belgacem, A., Beghdad-Bey, K., & Nacer, H. (2018, October). Task scheduling optimization in cloud based on electromagnetism metaheuristic algorithm. In 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS) (pp. 1-7). IEEE.

[12] Aziza, H., & Krichen, S. (2018). Bi-objective decision support system for task-scheduling based on genetic algorithm in cloud computing. Computing, 100(2), 65-91.

[13] Li, K., Xu, G., Zhao, G., Dong, Y., & Wang, D. (2011, August). Cloud task scheduling based on load balancing ant colony optimization. In 2011 sixth annual ChinaGrid conference (pp. 3-9). IEEE.

[14] Chen, X., & Long, D. (2019). Task scheduling of cloud computing using integrated particle swarm algorithm and ant colony algorithm. Cluster Computing, 22(2), 2761-2769.

[15] Liu, C. Y., Zou, C. M., & Wu, P. (2014, November). A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing. In 2014 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (pp. 68-72). IEEE.

[16] Tsai, C. W., Huang, W. C., Chiang, M. H., Chiang, M. C., & Yang, C. S. (2014). A hyper-heuristic scheduling algorithm for cloud. IEEE Transactions on Cloud Computing, 2(2), 236-250.

[17] Malik, R. F., Rahman, T. A., Hashim, S. Z. M., & Ngah, R. (2007). New particle swarm optimizer with sigmoid increasing inertia weight. International Journal of Computer Science and Security, 1(2), 35-44.

[18] Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S., & Abraham, A. (2011, October). Inertia weight strategies in particle swarm optimization. In 2011 Third world congress on nature and biologically inspired computing (pp. 633-640). IEEE.

[19] Mafarja, M., Aljarah, I., Faris, H., Hammouri, A. I., Ala'M, A. Z., & Mirjalili, S. (2019). Binary grasshopper optimisation algorithm approaches for feature selection problems. Expert Systems with Applications, 117, 267-286.

[20] Roth, G., & Dicke, U. (2005). Evolution of the brain and intelligence. Trends in cognitive sciences, 9(5), 250-257.

[21] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience, 41(1), 23-50.

[22] Yildirim, G., Alatas, B. New adaptive intelligent grey wolf optimizer based multi-objective quantitative classification rules mining approaches. J Ambient Intell Human Comput (2021). https://doi.org/10.1007/s12652-020-02701-9

[23] Koyuncu, H. GM-CPSO: A New Viewpoint to Chaotic Particle Swarm Optimization via Gauss Map. Neural Process Lett 52, 241–266 (2020). https://doi.org/10.1007/s11063-020-10247-2Electromagnetic Fields (300 Hz to 300 GHz), Environmental Health Criteria 137, World Health Organization, Geneva, Switzerland, 1993.

[24] Khishe M, Nezhadshahbodaghi M., Mosavi M.R., Martín D., "A Weighted Chimp Optimization Algorithm," in IEEE Access, vol. 9, pp. 158508-158539, 2021, doi: 10.1109/ACCESS.2021.3130933.

BIOGRAPHIES

EMRULLAH GUNDUZALP graduated from Firat University, Department of Computer Engineering in 2007 with a bachelor's degree. In 2018, he completed his master's degree in Firat University. He worked at the General Directorate of Land Registry and Cadastre between 2007-2009. Since 2009, he has been working as a computer engineer at the 9th Regional Directorate of DSI, IT department. His research interests are the propagation of electromagnetic waves, machine learning, and optimization.

GUNGOR YILDIRIM received the B.Sc. degree from Firat University, Turkey, in electrical and electronic engineering. He received the M.Sc. and the Ph.D. degrees from computer engineering of Firat University, in 2012, and 2017, respectively. Currently, he is an assistant professor at the Department of Computer Engineering at Firat University. He worked as a project and control engineer in private sectors and a public institution (DSI). He served as the head of the electric program at Tunceli MYO for three years. In 2014, he received the Award of Appreciation in DSI. His research interests include wireless systems, wireless sensor networks and IoT systems.

YETKIN TATAR received the B.Sc. degree from EDMMA in 1974 and the M.Sc. and D.Sc. degrees in electrical and electronic engineering from Firat University, Turkey, in 1984 and 1994, respectively.

He worked as a Professor with the Department of Computer Engineering, Firat University. His research areas are wireless sensor networks, computer networks, and network security.