Sohag Journal of Sciences
An International Journal

Task Scheduling Optimization in cloud computing by Cuckoo Search Algorithm

Ahmed Y. Hamed[1], *M. Kh. Elnahary*[1], *and Hamdy H. El-Sayed*[1]

[1]Faculty of Computers and Artificial Intelligence, Department of Computer Science, Sohag University, Sohag, 82524, Egypt.

Received: 9 Apr. 2022, Revised: 22 May 2022, Accepted: 29 May 2022.
Published online: 1 Spt. 2022.

Abstract: In cloud computing systems, task scheduling is crucial. Task scheduling cannot be done based on a single criterion but rather on rules and regulations which can be referred to as an agreement between cloud customers and providers. This agreement is nothing more than the user's desire for the providers to offer the kind of service that they expect. Providing high-quality services to consumers under the deal is a critical duty for providers, who must also manage many responsibilities. The task scheduling problem may be considered the search for an ideal assignment or mapping of a collection of subtasks of distinct tasks across the available set of resources to meet the intended goals for tasks. This paper proposes an efficient scheduling task algorithm based on the cuckoo search algorithm in cloud computing systems. By applying it to three cases, we evaluate the performance of our algorithm. The findings suggest that the proposed strategy successfully achieved the best solution in makespan, speedup, efficiency, and throughput.

Keywords: Heterogeneous resources, Cuckoo search algorithm, Task scheduling, Cloud Computing.
.

1 Introduction

There is no specific definition of cloud, but we may describe cloud in various ways and multiple approaches. Cloud computing is a type of supercomputing that is accessible over the internet. It is a shared infrastructure that connects large system pools using various methods such as distributed computing, virtualization, etc. It provides customers with a range of storage, networking, and computing resources in the cloud computing environment over the internet, allowing users to store a large amount of information and access a large amount of computational power using their computers [1]. The primary purpose of cloud computing is to manage computing power, storage, multiple platforms, and services assigned to external users on-demand over the internet. Cloud computing is a fast-evolving computation paradigm to relieve cloud users of the burden of managing hardware, software, networks, and data resources and moving them to cloud service providers. Clouds offer a wide range of resources, including computing platforms, data centres, storage, networks, firewalls, and software in services. At the same time, it provides methods for controlling these resources so that cloud users may use them without experiencing any performance issues. Cloud Computing Services are classified into three types based on the abstraction level and the service model of the provider: (1) Infrastructure as a Service(IaaS), (2) Platform as a Service(Paas), and (3) Software as a Service(SaaS). The fundamental qualities of cloud computing are distribution, virtualization, and elasticity. Virtualization is a crucial aspect of the cloud. Virtualization is supported by the vast majority of software and hardware. We can virtualize various components, such as hardware, software, storage, and operating systems, and manage them in a cloud platform [1].

To solve the task scheduling problem satisfactorily, we have presented an efficient method based on the cuckoo search algorithm called the efficient cuckoo search (ECS) algorithm to decrease the makespan and maximize the speedup, efficiency, and throughput.

The paper is organized as follows: The notations are presented in section 2. Related work is presented in Section 3. problem description is given in Section 4. The cuckoo search algorithm with levy flights is provided in Section 5 and Section 6. Section 7 describes the ECS approach. The evaluation of the proposed algorithm is presented in section 8. Section 9 concludes and offers future work.

* Corresponding author E-mail: mk409055@gmail.com

2 Notation

DG	is the graph of tasks
TS_i	is the task i
VM_i	is the virtual machine i
NVM	is the virtual machine's number
NTS	is the number of tasks
$COMC(TS_i, TS_j)$	is the communication cost between TS_i and TS_j
$St\_Time(TS_i, VM_j)$	is the start time of task i on a VM_j
$Ft\_Time(TS_i, VM_j)$	is the finish time of task i on a VM_j
$Re\_Time(VM_i)$	is the VM's ready time i
DLT	is a list of tasks arranged in topological order of DAG
$Da\_Arriv(TS_i, VM_j)$	is the time of task's i data arrival to VM_j

3 Related work

Cloud computing is a new technology that allows consumers to pay as they go and offers excellent performance. Cloud computing is also a heterogeneous system that stores many application data. It is accepted that optimizing the transferring and processing time is critical to an application program when scheduling some intensive data or computing an intensive application. The authors develop a task scheduling model and propose a particle swarm optimization (PSO) method based on this study's small position value rule [2] to reduce processing costs.

Cloud computing has lately experienced rapid growth and has emerged as a commercial reality in information technology. Cloud computing is a supplement, consumption, and delivery model for internet-based IT services charged per usage. The scheduling of cloud services influences the cost-benefit of this computing paradigm by service providers to users. Tasks should be scheduled efficiently in such a case so that the execution cost and time are decreased. In this research [3], the authors suggested a meta-heuristic-based scheduling method that reduces execution time and cost. An enhanced genetic algorithm is created by combining two existing scheduling methods for scheduling activities while considering their computational complexity and computing capability of processing elements.

The next generation of cloud computing will survive on the efficiency with which infrastructure is built and available resources are actively exploited. One of the primary issues in Cloud computing is load balancing, which distributes the dynamic workload over numerous nodes to guarantee that no one resource is either overburdened or underused. This is an optimization challenge, and a competent load balancer should adjust its method to the changing environment and job kinds. The Genetic Algorithm is used in this research [4] to suggest a unique load balancing approach (GA).

In cloud computing, job scheduling is an NP-hard optimization issue. Load balancing of non-preemptive independent jobs on virtual machines (VMs) is a critical component of cloud task scheduling. When specific VMs are overburdened with tasks to complete and the remaining VMs are underloaded, the load must be balanced to achieve optimal machine usage. In this research [5], the authors presented a new technique called honey bee behaviour inspired load balancing (HBB-LB), which attempts to produce a well-balanced load among virtual machines to maximize throughput. The suggested method additionally balances the priority of jobs on the computers such that the amount of waiting time for tasks in the queue is kept to a minimum.

Scheduling directed acyclic graph (DAG) tasks to minimize makespan has become a key topic in a range of applications on heterogeneous computing systems, including considerations regarding task execution order and task-to-processor mapping. The chemical reaction optimization (CRO) approach has recently proven valuable in various sectors. This paper [6] creates an enhanced hybrid version of the HCRO (hybrid CRO) approach to solve the DAG-based job scheduling issue. The CRO technique is combined with unique heuristic approaches in HCRO, and a new selection strategy is given. This study makes the following contributions in particular. (1) A Gaussian random walk technique is given to find the best local candidate solutions. (2) To ensure that our HCRO algorithm can escape from local optima, we adopt a left or right rotating shift approach based on maximum Hamming distance. (3) To preserve molecular diversity, a novel selection technique based on the normal distribution and a pseudo-random shuffling approach are proposed. Furthermore, an exclusive-OR (XOR) operator is placed between two strings to decrease the possibility of cloning before new molecules are created.

Task scheduling is one of the most important problems in heterogeneous cloud computing systems when high efficiency is required. Because task scheduling is a Nondeterministic Polynomial (NP)-hard issue, various evolutionary methods have been developed to address it. Because population-based algorithms have a slow convergence rate, they are combined with local search algorithms. Thus, this study [7] suggests a hybrid particle swarm optimization and hill-climbing method to improve the task scheduling makespan.

4 Problem Description

The task scheduling in cloud computing is represented as a Graph with NTS tasks (TS1, TS2, TS3, ..., TSNTS). Each task represents a task with DG and E-directed edges, signifying a portion of the tasks' requests

[8]. Each task means an instruction that might be performed sequentially on the same virtual machine alongside other instructions; it contains one or more inputs. The task an exit or entry task is triggered to execute based on the availability of the inputs. A precedence-constrained partial request result ($TS_i \rightarrow TS_j$), i.e., TS_i precedes TS_j in the process of execution. The execution time of a task TS_i is denoted by (TS_i) weight. Let COMC(TS_i, TS_j) be the cost of communication of an edge, and it will be equal to zero if TS_i and TS_j are scheduled on the same virtual machine. Start and finish times are denoted by St_Time(TS_i, VM_j) and Ft_Time(TSi, VMj), respectively [8]. The Da_Arriv of TS_i at virtual machine VM_j is given by:

$$\text{Da\_Arriv}(TS_i, VM_j) = \max\{\text{Ft\_Time}(TS_k, VM_j) + \text{COMC}(TS_i, TS_k)\} \quad (1)$$

Where k = 1.2, ..., number of Parents

The task scheduling issue in cloud computing may be characterized as finding the optimal assignment or schedule of the start times of the provided tasks on virtual machines. The scheduled length (completion time) and execution cost are reduced while keeping precedence constrained. The completion time is defined as the schedule length or makespan computed by:

$$\text{Completion Time} = \max(\text{Ft\_Time}(TSi, VMj)) \quad (2)$$

$$\text{Ft\_Time}(TSi, VMj) = \text{St\_Time}(TSi, VMj) + WTij \quad (3)$$

Where i = 1.2. ...., NTS, and j = 1,2, …NVM

Algorithm 1: To find the schedule length [8]

```
Input the schedule of tasks
Re_Time[$VM_j$] = 0      where    j = 1, 2, ……NVM.
For i = 1 : NTS
{
From DLT take the first task $TS_i$ to be executed and remove it from DLT.
 For j = 1 : NVM
  {
   If $TS_i$ is scheduled to virtual machine $VM_j$
    St_Time($TS_i$, $VM_j$) = max{Re_Time($VM_j$),Da_Arriv($TS_i$, $VM_j$)}
    Ft_Time($TS_i$, $VM_j$) = St_Time($TS_i$, $VM_j$) + WT($TS_i$,$VM_j$)
    Re_Time($VM_j$) = Ft_Time($TS_i$, $VM_j$)
   End If
  }
}
Schedule length = max(Ft_Time)
```

5 Levy Flights and Cuckoo Behavior

5.1 Cuckoo Breeding Habits

Cuckoos are intriguing birds, not just for their beautiful calls but also for their aggressive breeding method. Brood parasitism is classified into three types: intraspecific brood parasitism, cooperative breeding, and nest takeover. Some species, such as ani and Guiro cuckoos, deposit their eggs in communal nests, albeit they may remove the eggs of others to maximize the hatchability of their eggs [9]. Several species practise obligate brood parasitism by depositing their eggs in the nests of other host birds (often other species). Some host birds may engage in confrontation with invading cuckoos. If a host bird realizes that the eggs are not its own, it will either discard the foreign eggs or depart its nest and create a new one elsewhere. Some cuckoo species, like the New World brood-parasitic Tapera, have developed where female parasitic cuckoos are frequently highly specialized in the colour and pattern mimicry of a few selected host species' eggs. This minimizes the likelihood of their eggs being abandoned, increasing their reproductivity. Furthermore, the timing of egg-laying in several species is astonishing. In general, cuckoo eggs hatch significantly sooner than host eggs. Parasitic cuckoos frequently seek nests where the host bird has recently placed its eggs. When the first cuckoo chick hatches, its initial inclination is to evict the host eggs by blindly shoving the eggs out of the nest, increasing the cuckoo chick's portion of food given by its host bird. According to research, a cuckoo chick may mimic the call of host chicks to obtain access to additional feeding opportunities [9].

5.2 Levy Flights

On the other hand, various research has shown that the flight behavior of many insects and animals has demonstrated the typical L´evy flights characteristics [9]. According to recent research by Reynolds and Frye, fruit flies, or Drosophila melanogaster, investigate their environment utilizing a succession of straight flight pathways broken by a rapid 90o turn, resulting in a Levy-flying-style irregular scale-free search pattern. Following that, similar behaviour has been used to optimize and optimal search, with preliminary results demonstrating promising capabilities. Human behaviour studies, such as the Ju/'hoansi hunter-gatherer feeding habits, also reveal the common trait of Levy flights. Levy flights can even be associated with light [9].

6 Cuckoo Search

We now employ three idealized principles to describe our new Cuckoo Search for simplicity: 1) Each cuckoo lays one egg at a time and deposits it in a randomly selected nest. 2) The best nests with high-quality eggs will be passed down to future generations. 3) The number of possible host nests is fixed, and the egg placed by a cuckoo is detected with a probability pa ∈ [0, 1] by the host bird. The host bird can either discard the egg or depart the nest and create a new one in this instance. This second assumption can be approximated for simplicity by the proportion pa of the n nests replaced by new nests (with new random solutions). The quality or fitness of a solution to a maximizing issue might be proportional to the value of the objective function. Other kinds of fitness, such as the fitness function in genetic algorithms, can be described similarly. For simplicity, we may use the following simple representations: each egg in a nest symbolizes a solution, and a cuckoo egg indicates a new solution; the goal is to employ the latest and potentially better solutions (cuckoos) to replace a not-so-good solution in the nests. This approach may be expanded to a more sophisticated scenario where each nest has many eggs, indicating a set of solutions. We will choose the most straightforward technique for this study, with each nest containing simply one egg [9]. the newly generated solutions H_i^{s+1} is

$$H_i^{s+1} = H_i^{s} + \beta \oplus \text{Levy}\ (\gamma) \qquad (4)$$

Where $\beta > 0$ is the step size that should be connected to the problem of interests' scales. Most of the time, we can use $\beta = 1$. The preceding equation is the stochastic equation for a random walk. A random walk, in general, is a Markov chain whose future status/position is determined only by the present location (the first component in the above equation) and the transition probability (the second term). The term "product" refers to entrywise multiplications. This entrywise product is similar to those used in PSO, but the random walk through Levy flight is more efficient in exploring the search space in the long run since its step length is significantly greater. The Levy flight gives a random walk, with the arbitrary step length selected from a Levy distribution [9].

$$\text{Levy} \sim q = s^{-\gamma} \qquad (5)$$

It has an infinite variance and a zero mean. The steps in this case effectively constitute a random walk process with a power-law step-length distribution and a long tail. Levy should stroll around the best solution obtained so far to produce some new solutions, which will speed up the local search. However, a significant percentage of the latest solutions should be created by field randomization. Their locations should be sufficiently distant from the best solution to ensure that the system is not caught in a local optimum. There is some resemblance between CS and hill-climbing combined with some large-scale randomness from a cursory examination. There are, nevertheless, some essential variances. To begin, CS is a population-based algorithm, similar to GA and PSO, but it employs some form of elitism and selection, identical to harmony search. Second, randomization is more efficient because the step length is heavy-tailed, and any vast step is possible. Third, the number of parameters to be modified is smaller than that of GA and PSO, making it potentially more general in adapting to a broader class of optimization issues. Furthermore, because each nest might represent a collection of solutions, CS can be extended to the sort of meta-population algorithm [9].

Algorithm 2: Cuckoo Search via Levy Flights [9]

```
Objective function G(y), y = $(y_1, ..., y_d)^T$
Create a starting population of n host nests $y_i$ (i = 1, 2, ..., n)
while (s <Max_Generation or criteria for stopping)
        By Levy, flights get a cuckoo randomly and assess its quality/fitness $FT_i$
        Choose a nest at random from n (say, j).
        if ($FT_i > FT_j$),
            replace j with the new obtained solution;
        end if
        pa of the worst nests are abandoned, and new ones are constructed.
        Save the best solutions or nests with high-quality solutions.
      Sort the solutions and pick the best one right now.
end while
Visualization of postprocess results
```

7 The ECS Approach

It is clear that the representation of a vector in the cuckoo search algorithm is a continuous value form, so we will use the five methods to convert these continuous values to discrete values. The first is the Smallest Position Value (SPV) rule [10], and the second is the Largest Position Value (LPV) rule [11] and by using the modulus function with the number of virtual machines and increasing the value by one, as shown in Figure 1 and Figure 2.

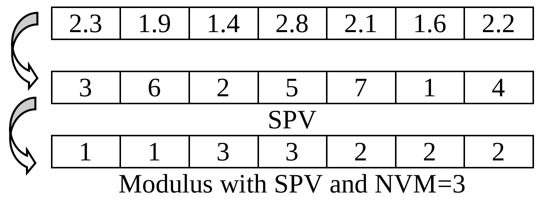

Figure 1. An example of converting with the SPV rule

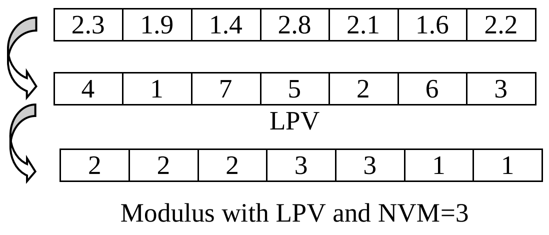

Figure 2. An example of converting with the LPV rule

3	1	3	2	2	1	2

Figure 3. An example proposed schedule

Algorithm 3: The function that converts a continuous value to a discrete value

```
Function converting(s)
Ran=random number between   [1…5]
If (Ran == 1)
        Use method of SPV rule as shown in Figure 1
Else if (Ran == 2)
        Use method of LPV rule as shown in Figure 2
Else if (Ran == 3)
        Use round nearest function
Else if (Ran == 4)
        Use floor nearest function
Else
        Use ceil nearest function
End if
End function
```

Algorithm 4: ECS

```
Input the DAG with computation and communication costs
Create a starting population of n host nests $y_i$ (i = 1, 2, ..., n)
Convert the starting population by using Algorithm 3
Calculate the schedule length by using Algorithm 1
while (s <Max_Generation or criteria for stopping)
   By Levy, flights get a cuckoo randomly and convert it by using Algorithm 3, then assess its quality or fitness $FT_i$ by using Algorithm 1
      Choose a nest at random from n (say, j).
      if ($FT_i > FT_j$),
         replace j with the new obtained solution;
      end if
      pa of the worst nests are abandoned, and new ones are constructed.
      Save the best solutions or nests with high-quality solutions.
   Sort the solutions and pick the best one right now.
end while
Visualization of postprocess results
```

8 Evaluation of the ECS

We show the performance of the ECS by applying it to three cases. The first case of 11 tasks and three heterogeneous virtual machines. The second case consists of 11 tasks and three heterogeneous virtual machines. The third one consists of three heterogeneous virtual machines and 10 tasks.

$$\text{Speedup} = \min_{VM_j} \left(\sum_{TS_i} \frac{WT_{i,j}}{\text{schedule length}} \right) \quad (12)$$

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{NVM}} \quad (13)$$

$$\text{Throughput} = \frac{\text{NTS}}{\text{Schedule Length}} \quad (14)$$

8.1 Case 1

We consider a case of 11 tasks {TS0, TS1, TS2, TS3, TS4, TS5, TS6, TS7, TS8, TS9, TS10} to be executed on three heterogeneous virtual machines {VM1, VM2, VM3}. The cost of running every task on different virtual machines is shown in Table 1 [12]. Table 2 represents each task's start time and finish time on other virtual machines and the schedule obtained by ECS. The results obtained by the ECS are compared with those obtained by Upward Rank [13], Downward Rank [13], Level Rank [13], BGA [14], and GA_DE_HEFT [12].

Figure **3**, Figure 4, Figure 5, and Figure 6 represent the results obtained by the ECS, Upward Rank, Downward Rank, Level Rank, BGA, GA_DE_HEFT in terms of makespan, speedup, efficiency, and throughput.

Table 1. Computation Cost for case 1

Task	VM_1	VM_2	VM_3
TS_0	9	11	10
TS_1	11	7	9
TS_2	8	6	4
TS_3	6	5	7
TS_4	9	17	10
TS_5	7	5	9
TS_6	12	15	9
TS_7	17	12	13
TS_8	8	12	10
TS_9	16	15	14
TS_{10}	11	10	12

Table 2. Schedule obtained by ECS for case 1

	VM_1		VM_2		VM_3	
	Start	Finish	Start	Finish	Start	Finish
TS_0	0	9	-	-	-	-
TS_1	-	-	21	28	-	-
TS_2	-	-	-	-	23	27
TS_3	9	15			-	-
TS_4	-	-	35	52	-	-
TS_5	-	-	-	-	27	36
TS_6	15	27	-	-	-	-
TS_7	-	-	53	65	-	-
TS_8	43	51	-	-	-	-
TS_9	27	43	-	-	-	-
TS_{10}	-	-	66	76	-	-

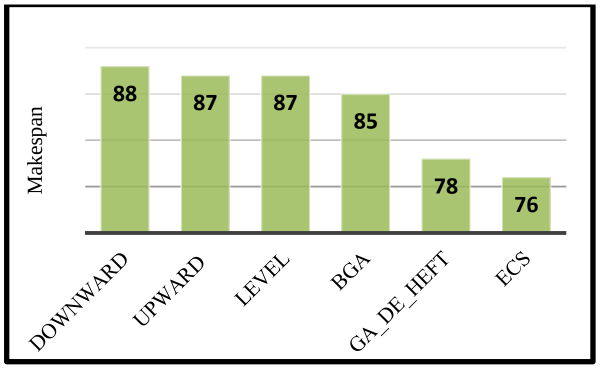

Figure 3. comparison of makespan for case 1

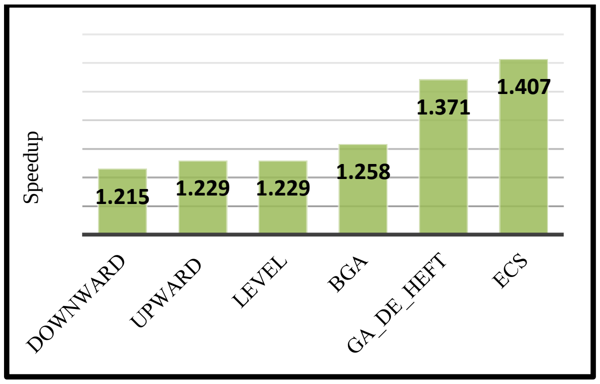

Figure 4. comparison of speedup for case 1

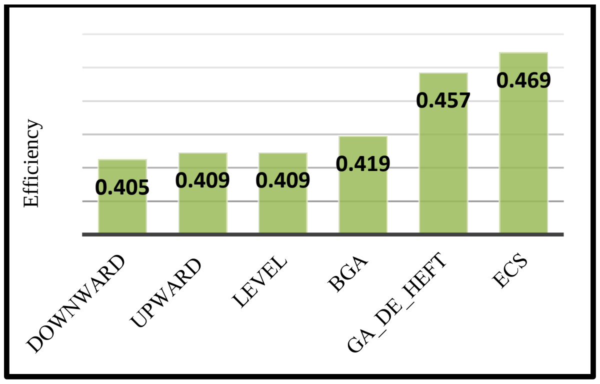

Figure 5. comparison of efficiency for case 1

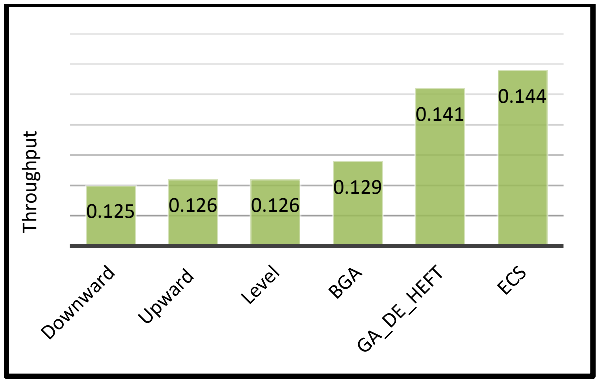

Figure 6. comparison of throughput for case 1

8.2 Case 2

We consider a case of 11 tasks {TS1, TS2, TS3, TS4, TS5, TS6, TS7, TS8, TS9, TS10, TS11} to be executed on three heterogeneous virtual machines {VM1, VM2, VM3}. The cost of running every task on different virtual machines is shown in

Table **3** [15].

Table **4** represents each task's start time and finish time on other virtual machines and the schedule obtained by ECS. The results obtained by the ECS are compared with those obtained by HEFT [15], CPOP [15], and MHEFT [15]. Figure 7, Figure 8, Figure 9, and Figure 10 represent the results obtained by the ECS, HEFT, CPOP, and MHEFT in terms of makespan, speedup, efficiency, and throughput.

Table 3. Computation Cost for Case 2

Task	VM_1	VM_2	VM_3
TS_1	16	19	27
TS_2	18	15	13
TS_3	21	12	22
TS_4	15	13	11
TS_5	22	19	20
TS_6	13	09	11
TS_7	8	11	16
TS_8	14	23	10
TS_9	28	32	12
TS_{10}	15	13	09
TS_{11}	14	16	22

Table 4. Schedule obtained by ECS for case 2

	VM_1		VM_2		VM_3	
	Start	Finish	Start	Finish	Start	Finish
TS_1	0	16	-	-	-	-
TS_2	-	-	-	-	33	46
TS_3	-	-	36	48	-	-
TS_4	38	53	-	-	-	-
TS_5	16	38	-	-	-	-
TS_6	-	-	72	81	-	-
TS_7	-	-	-	-	57	73
TS_8	-	-	-	-	73	83
TS_9	-	-	-	-	83	95
TS_{10}	-	-	94	107	-	-
TS_{11}	-	-	107	123	-	-

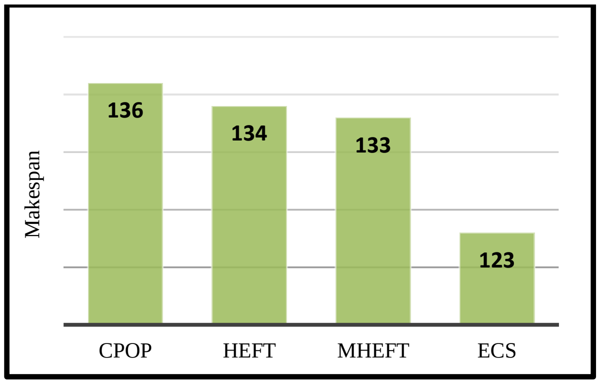

Figure 7. comparison of makespan for case 2

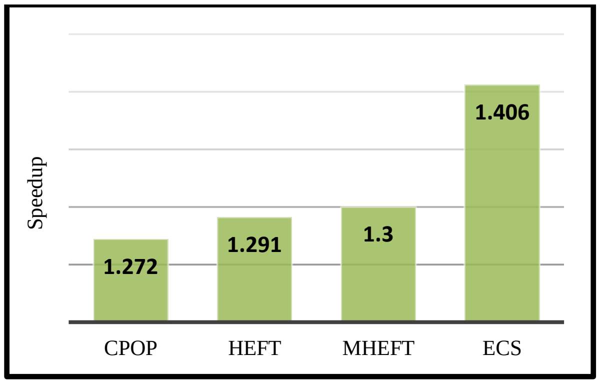

Figure 8. comparison of speedup for case 2

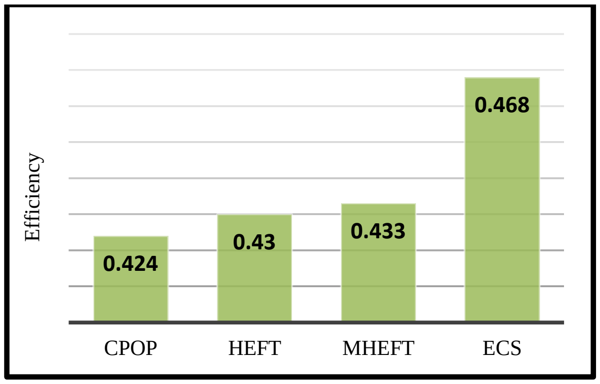

Figure 9. comparison of efficiency for case 2

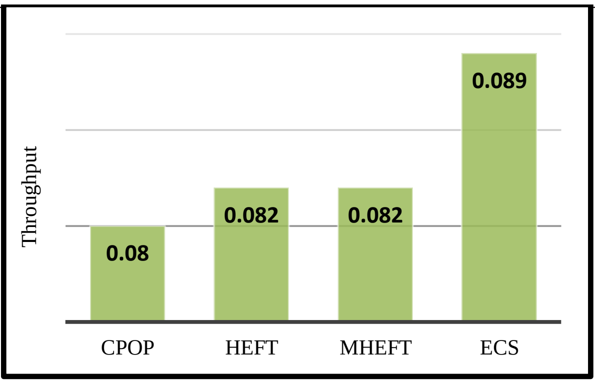

Figure 10. comparison of throughput for case 2

8.3 Case 3

We consider a case of 10 tasks {TS0, TS1, TS2, TS3, TS4, TS5, TS6, TS7, TS8, TS9} to be executed on three heterogeneous virtual machines {VM1, VM2, VM3}. The cost of running every task on different virtual machines is shown in

Table **5** [6]. Table 6 represents each task's start time and finish time on other virtual machines and the schedule obtained by ECS. The results obtained by the ECS are compared with those obtained by HCRO [6]. Figure 11, Figure 12, Figure 13, and Figure 14 represent the results obtained by the ECS and HCRO in terms of makespan, speedup, efficiency, and throughput.

Table 5. Computation Cost for Case 3

Task	VM_1	VM_2	VM_3
TS_0	10	11	11
TS_1	9	10	8
TS_2	8	6	8
TS_3	10	10	9
TS_4	13	12	13
TS_5	3	2	4
TS_6	10	8	9
TS_7	2	2	2
TS_8	18	17	16
TS_9	15	14	14

Table 6. Schedule obtained by ECS for case 3

	VM_1		VM_2		VM_3	
	Start	Finish	Start	Finish	Start	Finish
TS_0	-	-	-	-	0	11
TS_1	13	22	-	-	-	-
TS_2	-	-	-	-	11	19
TS_3	-	-	-	-	19	28
TS_4	-	-	12	24	-	-
TS_5	-	-	25	27	-	-
TS_6	27	37	-	-	-	-
TS_7	-	-	30	32	-	-
TS_8	-	-	-	-	28	44
TS_9	-	-	-	-	44	58

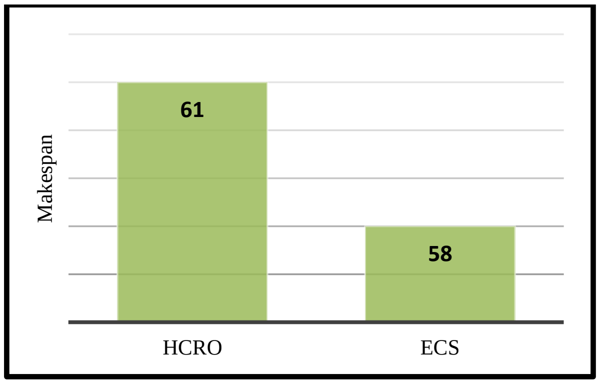

Figure 11. comparison of makespan for case 3

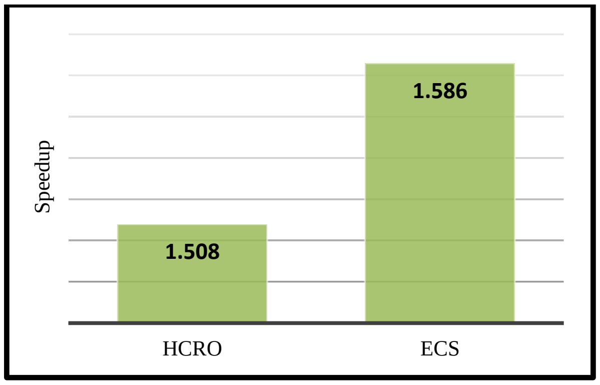

Figure 12. comparison of speedup for case 3

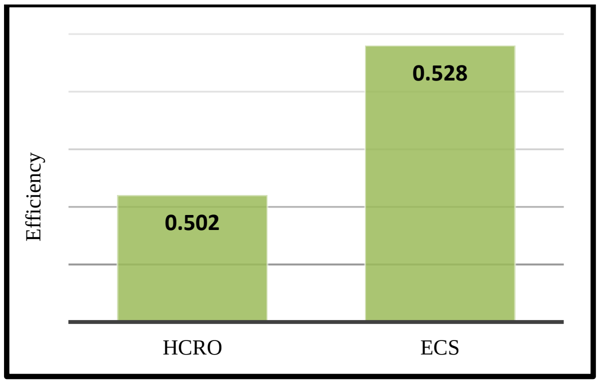

Figure 13. comparison of efficiency for case 3

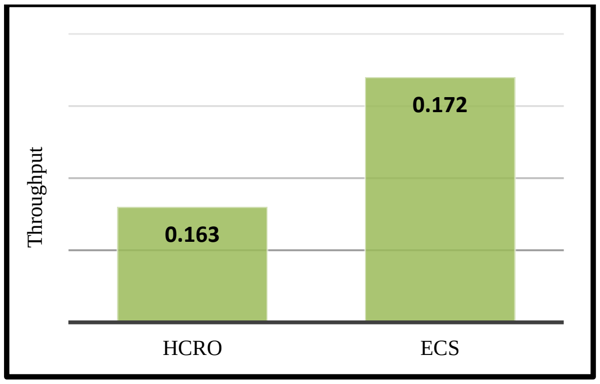

Figure 14. comparison of throughput for case 3

9 Conclusion and Future Work

The proposed efficient cuckoo search algorithms allocate or schedule subtasks to available virtual machines in a cloud computing environment. According to the obtained results on DAGs of different cases, the efficient cuckoo search algorithms are significantly more effective than other algorithms in terms of makespan, speedup, efficiency, and throughput. We compared the results of ECS, Upward Rank, Downward Rank, Level Rank, BGA, GA_DE_HEFT, it is clear that the length of ECS's schedule was less than that of Upward Rank, Downward Rank, Level Rank, BGA, GA_DE_HEFT as shown in Figure 3. Also, we compared the results of ECS, HEFT, CPOP, and MHEFT, it is clear that the length of ECS's schedule was less than that of HEFT, CPOP, and MHEFT as shown in Figure 7. In addition, the ECS results were better than those found by HCRO as shown in Figure 11. In the future, we will develop an algorithm based on DAGs by considering the load balancing of the resources.

References

[1] Singh, R.M; Paul, S. and Kumar, A.; Task Scheduling in Cloud Computing : Review, *International Journal of Computer Science and Information Technologies.*, 2014, **5 (6)**, 7940–7944.

[2] Guo, L.; Zhao, S.; Shen, S. and Jiang, C.; Task scheduling optimization in cloud computing based on heuristic Algorithm, *Journal of Networks*., 2012**, 7 (3)**, 547–553.

[3] Kaur, S. and Verma, A.; An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment, *International Journal of Information Technology and Computer Science*., 2012**, 4 (10)**, 74–79.

[4] Dasgupta, K.; Mandal, B.; Dutta, P.; Mandal, K.J and Dam, S.; A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing, *Procedia Technology*., 2013, **10**, 340–347.

[5] Dhinesh Babu D.L and Venkata Krishna, P.; Honey bee behavior inspired load balancing of tasks in cloud computing environments, *Applied Soft Computing*., 2013**, 13**, 2292–2303.

[6] Xu, Y.; Li, K.; He, L.; Zhang, L. and Li, K.; A Hybrid Chemical Reaction Optimization Scheme for Task Scheduling on Heterogeneous Computing Systems, *IEEE Transactions on Parallel and Distributed Systems.*, 2015**, 26 (12)**, 3208–3222.

[7] Dordaie, N. and Navimipour, J.N; A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments, *ICT Express*., 2018**, 4**, 199–202.

[8] Hamed, Y.A and Alkinani, M.H; Task scheduling optimization in cloud computing based on genetic algorithms, *Computers, Materials and Continua*., 2021**, 69 (3)**, 3289–3301.

[9] Yang, S.X and Deb, S.; Cuckoo search via Lévy flights, *2009 World Congress on Nature and Biologically Inspired Computing (NABIC)*., 2009, 210–214.

[10] Dubey, I. and Gupta, M.; Uniform mutation and SPV rule based optimized PSO algorithm for TSP problem, *in Proc. of the 4th International Conference on Electronics and Communication Systems*., Coimbatore, India, 2017, 168–172.

[11] Wang, L.; Pan, Q. and Tasgetiren M.F; A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem, *Computers & Industrial Engineering*., 2011, **61 (1)**, 76-83.

[12] Kamalinia, A. and Ghaffari, A.; Hybrid Task Scheduling Method for Cloud Computing by Genetic and DE Algorithms, *Wireless Pers Commun*., 2017, **97**, 6301–6323.

[13] Topcuoglu, H.; Hariri, S. and Wu Y.M; Performance-effective and low-complexity task scheduling for heterogeneous computing, *IEEE Transactions on Parallel and Distributed Systems.*, 2002**, 13**, 260–274.

[14] Gupta, S.; Agarwal, G. and Kumar, V.; Task scheduling in multiprocessor system using genetic algorithm, *2010 2nd International Conference on Machine Learning and Computing*., 2010, 267–271.

[15] Dubey, K.; Kumar, M. and Sharma C.S; Modified HEFT Algorithm for Task Scheduling in Cloud Environment, *Procedia Computer Science*., 2018**, 125**, 725–732.