Volume: 14 Issue: 06 Pages: 5686 - 5695 (2023) ISSN: 0975-0290

Task Scheduling Optimization in Cloud Computing by Coronavirus Herd Immunity Optimizer Algorithm

Ahmed Y. Hamed
Faculty of Computers and Artificial Intelligence, Department of Computer Science, Sohag University, Sohag, 82524, Egypt
Email: ayhamedd@gmail.com
M. Kh. Elnahary
Faculty of Computers and Artificial Intelligence, Department of Computer Science, Sohag University, Sohag, 82524, Egypt
Email: mk409055@gmail.com
Hamdy H. El-Sayed
Faculty of Computers and Artificial Intelligence, Department of Computer Science, Sohag University, Sohag, 82524, Egypt
Email: hamdy2006x@gmail.com

----------**ABSTRACT**----------

Cloud computing is now dominant in high-performance distributed computing, offering resource polling and on-demand services over the web. So, the task scheduling problem in a cloud computing environment becomes a significant analysis space due to the dynamic demand for user services. The primary goal of scheduling tasks is to allocate tasks to processors to achieve the shortest possible makespan while respecting priority restrictions. In heterogeneous multiprocessor systems, task and schedule assignments significantly impact the system's operation. Therefore, the different processes within the heuristic-based scheduling task algorithm will lead to a different makespan on a heterogeneous computing system. Thus, a suitable algorithm for scheduling should set precedence efficiently for every subtask based on the resources required to reduce its makespan. This paper proposes a novel efficient scheduling task algorithm based on the coronavirus herd immunity optimizer algorithm to solve task scheduling problems in a cloud computing environment. The basic idea of this method is to use the advantages of meta-heuristic algorithms to get the optimal solution. We evaluate the performance of our algorithm by applying it to three cases. The collected findings suggest that the proposed strategy successfully achieved the best solution in terms of makespan, speedup, efficiency, and throughput compared to others. Furthermore, the results demonstrate that the suggested technique beats existing methods new genetic algorithm (NGA), proposed particle swarm optimization (PPSO), whale optimization algorithm (WOA), enhanced genetic algorithm for task scheduling (EGA-TS), gravitational search algorithm (GSA), genetic algorithm (GA), and hybrid heuristic and genetic (HHG) by 22.8%, 12.3%, 8.8%, 7.3%, 7.3%, 3.4%, and 3.4% respectively according to makespan.

Keywords -**Cloud Computing, Coronavirus Herd Immunity Optimizer Algorithm, Heterogeneous Processors, Task Scheduling.**

Date of Submission: March 20,2023 Date of Acceptance: April 26,2023

1. INTRODUCTION

As Internet access and extensive data become more freely available, cloud computing is becoming more prevalent in today's business environment. Compared to prior systems of distributed computing, for example, grid and cluster computing, cloud computing has created a more elastic and scalable method of delivering services to users. Consumers are not required to own the underlying technology and can use platforms and resources for computing on a pay-per-use basis. The basic idea behind cloud computing is to delegate computing work to a resource pool of many virtual machines or heterogeneous virtualized servers. Because cloud computing is a market-oriented utility, enhanced scheduling resources, which may support workflows, tasks, user applications, software, etc., are constantly required to maximize the users' and cloud providers' profits. Indeed, scheduling may directly impact a system's performance, such as operating cost, efficiency, and resource utilization, and it is seen as critical in cloud computing. Because virtual machines (VMs) may be given, assigned, and managed dynamically, cloud computing scheduling difficulties can be divided into two groups. The first is a virtual machine and mapping host that creates or migrates a virtual machine to a suitable host. The second is scheduling user-submitted tasks and mapping them to a set of available virtual machine resources [1].

To solve the scheduling task problem in a cloud computing environment, we have presented in this paper a new efficient approach based on the coronavirus herd immunity optimizer algorithm called the efficient

Volume: 14 Issue: 06 Pages: 5686 - 5695 (2023) ISSN: 0975-0290

coronavirus herd immunity optimizer algorithm (ECHIOA) to minimize the makespan of the user requests on the resources and maximize the speedup, efficiency, and throughput. In the coronavirus herd immunity optimizer algorithm, the representation of a vector is a continuous value, so we use five methods to convert the continuous value to a discrete value. We assess our algorithm's performance by running it through three cases. The results show that the proposed method found the best solutions faster and more efficiently than other algorithms regarding makespan, speedup, efficiency, and throughput.

The paper is organized as follows: The notations are presented in section 2. The related work is provided in Section 3. A description of the problem is given in Section 4. The coronavirus herd immunity optimizer algorithm is given in Section 5. Section 6 describes the ECHIOA approach. The results were obtained by applying ECHIOA and compared with the other results in Section 7. Finally, section 8 concludes and offers future work.

2. NOTATIONS

GT	It represents the graph of tasks
Ta_i	It represents the task i
Pr_i	It represents the processor i
MM	It represents the processor's number
NT	It represents the task's number
$C(Ta_i, Ta_j)$	The communication cost between Ta_i and Ta_j
$Start\_Time(Ta_i, Pr_j)$	It represents the task's start time i on a processor Pr_j
$Finish\_Time(Ta_i, Pr_j)$	It represents the task's finish time i on a processor Pr_j
$Ready\_Time(Pr_i)$	It represents the processor's ready time i
LT	It represents a list of jobs arranged in directed acyclic graph topological order
$Data\_Arrive(Ta_i, Pr_j)$	It represents the time of data arrival of task i to processor j

3. RELATED WORK

In recent years, there has been a lot of interest in cloud computing technologies, both in academics and business. The potential of cloud computing to supply worldwide information technology services such as platforms, core infrastructure, and applications to cloud consumers over the Internet has fueled its appeal. It also offers on-demand services and different price packages. On the other hand, cloud task scheduling is still nondeterministic polynomial (NP)-complete, and it has become more sophisticated owing to resource elasticity and on-demand customer application demand. To remedy this need, this work [2] proposes a modified harris hawks optimization (HHO) technique based on simulated annealing (SA) for cloud task scheduling. The SA is a local search method in the proposed harris hawks optimization simulated annealing (HHOSA) strategy to increase the pace of convergence and solution quality provided by the traditional HHO algorithm.

The demand for vast processing power and storage space has been increasing in many industries, and a novel cloud computing technology has been launched to meet this demand. Cloud computing technology has grown in popularity due to its capacity to provide these services efficiently and cost-effectively. With virtualization, information technology services have begun to transition to cloud computing, and virtualization has cleared the path for infinite resource availability. Because cloud computing is still in its early stages, additional research is required to realize its potential fully. More research is needed to determine how resources and tasks are assigned in a cloud setting, and it accounts for the quality of service (QoS) provided by cloud service providers. This study suggests utilizing the CloudSim toolkit to simulate the performance cost grey wolf optimization (PCGWO) method to optimize allocating resources and jobs in the cloud computing environment. The primary goal is to reduce cost and processing time in line with the target function [3].

Cloud computing is a technology that uses the Internet in which all programs and information are housed in a cloud made up of thousands of intricately connected machines. The main difficulty for cloud data centers is demonstrating how millions of requests from end users are examined and processed accurately and effectively. One of the most critical concerns in a distributed computing system is the load-balancing mechanism. Load-balancing solutions are required to boost the scalability and flexibility of cloud data centers. Because large-scale resources are available and a massive number of user requests are in the cloud computing load-balancing challenge, it is possible that many researchers evaluated and tackled it as an NP-hard problem. As a result, earlier researchers offered several heuristics algorithms, such as the imperialist competitive algorithm (ICA) and the firefly algorithm (FA), to overcome the abovementioned difficulty. However, ICA and FA could achieve an approximately satisfactory result in handling the cloud computing load-balancing problem; attaining a better result requires improvements in makespan, load balancing, central processing unit (CPU) time, stability, and planning speed. This study aims to provide a clever metaheuristic method based on ICA and FA to achieve the desired outcome. In addition, FA's local search capability can help to improve the ICA algorithm [4].

Cloud computing technology provides customers with pay-per-use computer resources without interfering with the underlying infrastructure. It is regarded as a vital repository of materials made available to consumers. Recently, there has been a significant increase in the desirability of cloud computing systems that rely on on-demand computing resources, a bill on a pay-as-you-go basis, and multiplex numerous users on the same physical infrastructure. Scheduling is a significant issue in cloud computing since a cloud provider must service various customers in a cloud environment. As a challenge to existing technologies, this proposal intends to establish an optimum job scheduling model in the cloud sector. The

Volume: 14 Issue: 06 Pages: 5686 - 5695 (2023) ISSN: 0975-0290

suggested model handles the job scheduling problem by employing an upgraded meta-heuristic algorithm known as the fitness rate-based rider optimization algorithm (FR-ROA), a more sophisticated version of the traditional rider optimization algorithm (ROA). The maximum makespan or completion time and the aggregate of the completion times of whole jobs are the constraints of objectives considered for optimum task scheduling because the suggested FR-ROA benefits from achieving convergence in a short time [5].

Cloud computing is a dynamic and diversified ecosystem that spans several geographies. It is made up of a plethora of tasks and computational resources. In the cloud, the scheduling of task algorithms plays a crucial role in determining the best virtual machine (VM) for a given job. The scheduling of the task algorithm is in charge of lowering the schedule's makespan. In recent years, nature-inspired algorithms have been used for work scheduling, outperforming traditional algorithms. The crow search algorithm (CSA) is presented in this study [6] for job scheduling in the cloud. Crows' food-gathering behaviors inspired it. In actuality, the crow continues to scan its surroundings for a better food source than its present one. As a result, the CSA selects an appropriate VM for the job and reduces the makespan.

One of the most challenging difficulties in cloud computing is efficient task scheduling. Because scheduling tasks is an NP-complete issue, finding the optimum solution is difficult, especially for large task sizes. Several jobs in the cloud computing environment may need to be efficiently scheduled on numerous virtual machines while reducing makespan and concurrently optimizing resource consumption. We introduce a unique hybrid antlion optimization method with elite-based differential evolution to handle multi-objective work scheduling issues in cloud computing settings. The multi-objective aspect of the problem in the proposed solution, which we term multi-objective antlion optimizer (MALO), stems from the necessity to decrease makespan while optimizing resource consumption concurrently. The antlion optimization algorithm was modified using a regional search strategy to boost its exploitation capabilities and prevent it from becoming stuck in local optima. Elite-based differential evolution is used [7].

Cloud computing is the rebel of worldwide networked resources and effortlessly shares data with users. With the extensive availability of network technology, user demands are increasing daily. The most significant issue in cloud technology now is task scheduling. Cargo position and task arrangement are critical criteria in the cloud domain that might guarantee QoS. In this study, the authors developed the optimal energy consumption reduction and makespan scheduling tasks using the local pollination-based gray wolf optimizer (LPGWO) algorithm. The gray wolf optimizer (GWO) and flower pollination algorithm (FPA) are merged and employed in the hybrid idea. In the presence of GWO, the optimal searching factor is employed to accelerate convergence. The FPA is used to distribute data to the following candidate packet solutions using the local pollination idea [8].

4. PROBLEM DESCRIPTION

The task scheduling model during this work is defined as distributed tasks to be implemented on processors. The processors may be different in general. A graph of tasks (GT) may be mapped to describe the problem structure. GT is a directed acyclic graph (DAG) of tasks: Ta_1, Ta_2, Ta_3, etc... Tan. Every node in the graph is termed a task. A task is assumed to be a series of instructions that must be carried out in a specific order by an assigned processor. A task (node) might have pre-demanded data (inputs) before implementation. The task may be activated to begin execution when all inputs have been received. These inputs are expected to be delivered after some other tasks are implemented, as these tasks evaluate them. We call this relying on task dependency. If a task depends on other tasks' data, we consider that task as the parent of the task, and the task is their child. A task with no parents is an entry task, while a task with no children is referred to as an exit task [9]. The time of execution of a task is the computation cost. Whenever the computation cost of a task Tai is represented by weight (Ta_i, Pr_j), the task graph also contains directed edges, representing a partial order of the tasks. The partial order establishes a DAG with a precedence constraint and indicates that if ($Ta_i \rightarrow Ta_j$), then Ta_j is a child. The child cannot begin until its parent Tai has finished. The weight on edge represents the communication cost between the tasks and is represented by C(Ta_i, Ta_j). The communication cost is considered only if Ta_i and Ta_j are allocated to different processors; otherwise, it equals zero. In that case, Ta_i and Ta_j are assigned to the same processor. If a node Ta_i is assigned to processor Pr_j, the start time of the task and the finish time are represented by Start_Time(Ta_i, Pr_j) and Finish_Time(Ta_i, Pr_j), respectively. After scheduling the tasks, the makespan is defined as the max {Finish_Time(Ta_i, Pr_j)} across all processors. The scheduling problem is to find a schedule of the tasks in the processors such that the makespan is decreased over possible schedules where the task dependency constraints are preserved. Task dependency constraints state that any task can't start until all parents have finished. Assume that Pr_j is the processor and that the KP^{th} parent task Ta_{kp} of task Ta_i is scheduled. The data-arrival of Ta_i at processor Pr_j is when the per-demanded data for the task execution becomes available. This is defined in [9] by the following: Data_Arrive(Ta_i, Pr_j) = max{Finish_Time(Ta_{kp} , Pr_j) + C(Ta_i , Ta_{kp})} where kp = 1, 2, 3,…. Parent_Number.

5. ALGORITHM FOR CORONAVIRUS HERD IMMUNITY OPTIMIZER

coronavirus herd immunity optimizer (CHIO) is depicted as a series of stages, which are extensively detailed below. The suggested optimization technique is based on herd immunity [10]. The method comprises six major phases, which are described below:

Step 1: Set the settings for coronavirus herd immunity and the problem of optimization. The issue of optimization is

Volume: 14 Issue: 06 Pages: 5686 - 5695 (2023) ISSN: 0975-0290

framed in the context of the objective function, as shown in this step:

$\min_{y} g(y)$ y ϵ [LOB, UPB] (1)

CHIO includes two parameters of control and four algorithmic techniques. The objective function g(y) is derived for the case y = (y_1, y_2,..., y_n), where y_i is the decision variable or gene indexed by n, and i is the overall total of genes in every individual. It is worth noting that the value range of every gene y_i is [LOB, UPB], with LOB and UPB being the lowest and upper boundaries of gene y_i. The four algorithmic parameters are:

W_0: This shows the number of first infected instances in this scenario, where it begins with one.

Maximum_Iter: This denotes the maximum number of iterations that may be performed.

Pops: This represents the population size.

n: This indicates the problem's dimensionality.

In this phase, the CHIO's two major control parameters must be initialized:

The rate of essential reproduction (BRr) governs the CHIO operators by propagating the viral pandemic among people.

The maximum age of infected cases (Maximum_age): It controls the state of infected patients, with those reaching their maximum age either recovering or dying.

Step 2: Create a population of herd immunity. Initially, CHIO produces a collection of individuals or instances as prominent as Pops at random or heuristically. The created cases are saved in the herd immunity population (HI_Pop) as a matrix of two dimensions and size (n * Pops), such as:

$$\text{HI\_Pop} = \begin{bmatrix} y_1^1 & \cdots & y_n^1 \\ \vdots & \cdots & \vdots \\ y_1^{Pops} & \cdots & y_n^{Pops} \end{bmatrix} \quad (2)$$

And every row j represents a case y_j, which is produced as follows:

y_i^j = LOB + (UPB-LOB) * P(0,1), where i=1, 2, . . .,n. Eq. 1 is used to compute every case's objective function or rate of immunity. Furthermore, the vector of status (ST) of length Pops for every case in HI_Pop is begun by either one infected case or zero susceptible cases. It is worth noting that the number in (ST) is chosen randomly and can be as high as W_0.

Step 3: Evolution of the coronavirus herd immunity. It is the primary improvement loop for CHIO. The gene (y_i^j) of case yj is either unchanged or modified by social distance according to three rules based on the fraction of the BRr, as follows:

$$y_i^j(u+1) = \begin{cases} y_i^j(u) & ran \geq BRr \\ L\left(y_i^j(u)\right) & ran < \frac{1}{3} * BRr \\ O\left(y_i^j(u)\right) & ran < \frac{2}{3} * BRr \\ E\left(y_i^j(u)\right) & ran < BRr \end{cases} \quad (3)$$

Where ran generates a number between 0 and 1 at random. The following are the three rules to consider:

In the infected case: the range of ran $\epsilon \left[0, \frac{1}{3}BRr\right)$, the value of the new gene of $y_i^j(u+1)$ is influenced by some social distance caused by the difference between the gene extracted from an infected case y^m and the present gene, such as:

$y_i^j(u+1)$= L(y_i^j(u) (4)

Where

L(y_i^j(u)) = y_i^j(u) + ran * (y_i^j(u) - y_i^c(u)) (5)

It should be noted that the value y_i^c(u) is picked at random from any infected case y^c based on the vector of status (ST) such that c = $\{i|ST_i = 1\}$.

In susceptible case: the range of ran $\epsilon \left[\frac{1}{3}BRr, \frac{2}{3}BRr\right)$, the value of the new gene y_i^j(u+1) is influenced by some social distance caused by the difference between the gene extracted from a susceptible case y^m and the present gene, such as:

y_i^j(u+1) = O(y_i^j(u)) (6)

Where

O(y_i^j(u)) = y_i^j(u) + ran * (y_i^j(u) -y_i^m(u)) (7)

It should be noted that the value y_i^m(u)is randomly distributed from any susceptible case y^m depending on the vector of status (ST), such that m = $\{i|ST_i = 0\}$.

In the immune case: the range of ran $\epsilon \left[\frac{2}{3}BRr, BRr\right)$,the value of the new gene y_i^j(u+1) is influenced by some social distance caused by the difference between the gene extracted from an immune case yv and the present gene, such as:

y_i^j(u+1) = E(y_i^j(u)) (8)

where

E(y_i^j(u)) = y_i^j(u) + ran * (y_i^j(u)-y_i^v(u)) (9)

It should be noted that the value y_i^v(u) is distributed from the best immune case yv depending on the vector of status (ST) in such a way that:

g(yv) = arg $\min_{j \sim \{k|ST_k = 2\}} g(y^j)$(10)

Step 4: Update the population of herd immunity. The rate of immunity g($y^j(u+1)$) of every created case $y^j(u+1)$ is computed, and the present case $y^j(u)$ is substituted with the created case $y^j(u+1)$, if better, such as g($y^j(u+1)$) < g($y^j(u)$) if ST_j=1, the age vector A_j increased by one.

For each case y^j, the status vector (ST_j) is updated depending on the threshold of the herd immune, which is calculated using the following equation:

$$\boldsymbol{ST_j} = \begin{cases} \mathbf{1} & \text{g}\left(y^j(u+1)\right) < \frac{\text{g}(y)^j(u+1)}{\Delta \boldsymbol{g}(y)} \wedge \boldsymbol{ST_j} = \mathbf{0} \wedge \boldsymbol{iscorona}(y^j(u+1)) \\ \mathbf{2} & \text{g}\left(y^j(u+1)\right) < \frac{\text{g}(y)^j(u+1)}{\Delta \boldsymbol{g}(y)} \wedge \boldsymbol{ST_j} = \mathbf{1} \end{cases} \quad \ldots(11)$$

where iscorona($y^j(u+1)$is a value in binary, and it is equal to one when the new case $y^j(u+1)$inherits a value from any infected case. The Δg(y) is the mean value of the population immunity rates, which is expressed as $\frac{\sum_{i=1}^{Pops} g(y_i)}{Pops}$. It should be noted that if the newly created individual rate of immunity is higher than the population's average rate of immunity, the immunity rate of the individuals in the population will be adjusted depending on the previously determined social distance. It suggests that our people are becoming more immune. We have reached the

Volume: 14 Issue: 06 Pages: 5686 - 5695 (2023) ISSN: 0975-0290

herd immunity threshold if the freshly produced population is immune to the pandemic.

Step 5: Cases of death If the present infected case (ST_j== 1) immunity rate g($y^j(u+1)$) doesn't improve after a given number of iterations as indicated by the parameter Maximum_age (i.e., Aj >= Maximum_ age). This case is declared dead. It is then recreated from scratch using $y_i^j(u+1)$= LOB + (UPB-LOB) * P(0,1), where i=1, 2, 3,...,n. In addition, Aj and STj are also set to zero. It can benefit diversifying and avoiding local optima and the present population.

Step 6: The criteria for stopping CHIO continue Steps 3–6 until the termination measure is met, which generally depends on whether the maximum number of iterations is met. In this situation, the population is mainly determined by the total number of immune and susceptible cases, and infected cases have vanished.

Algorithm CHIO

Initialize the parameters (Pops, Maximum_age, W0, Maximum_iter, BRr)
Generate the population By using y_i^j= LOB + (UPB-LOB) * P(0,1)
Compute the fitness function
Put STj=Aj =0 j= 1,2,…., Pops
While (t<= Maximum_iter)
For j=1: Pops
iscorona($y^j(u+1)$)= false
For i=1: N
If (ran < 1/3 * BRr)
Use Eq. 4, and Eq. 5
Iscorona($y^j(u+1)$)= true
Else if (ran < 2/3 * BRr)
Use Eq. 6 and Eq. 7
Else if (ran < BRr)
Use Eq. 8 and Eq. 9
Else
$y_i^j(u+1) = y_i^j(u)$
End if
End for
If ($g\left(y^j(u+1)\right)$<= $g\left(y^j(u)\right)$
$y^j(u)= y'^j(u+1)$
Else
Aj= Aj+1
End if
If ($g\left(y^j(u+1)\right) < \frac{g(y)^j(u+1)}{\Delta g(y)} \wedge ST_j = 0 \wedge iscorona(y^j(u+1))$
STj=Aj=1
End if
If ($g\left(y^j(u+1)\right) < \frac{g(y)^j(u+1)}{\Delta g(y)} \wedge ST_j = 1$)
STj=2
Aj=0
End if
If ((Aj >= Maximum_age) ∧ (STj==1))
$y_i^j(u+1)$= LOB + (UPB-LOB) * P(0,1)
STj=Aj=0
End if
End for
t=t+1
End while

6. THE PROPOSED ALGORITHM

We see that the representation of a vector is a continuous value, so we will use the smallest position value (SPV) rule [11] and the largest position value (LPV) rule [12]. After using the SPV or LPV, we will use the modulus function with the number of processors and increase the value by one, as shown in Fig. 1 and Fig. 2.

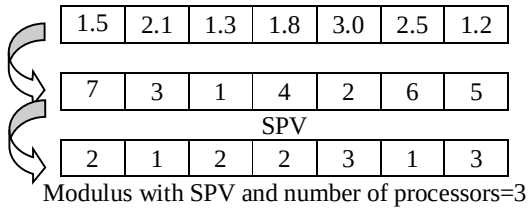


Figure 1: An example of the proposed schedule with the SPV rule

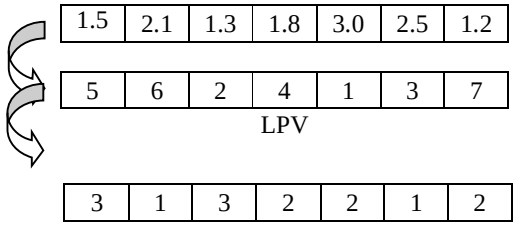


Figure 2: An example of the proposed schedule with the LPV rule

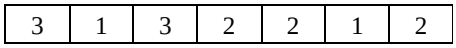


Figure 3: Proposed schedule

The tasks Ta_2 and Ta_6 are scheduled on processor one. The tasks Ta_4, Ta_5, and Ta_7 are scheduled on processor two. Finally, the tasks Ta_1 and Ta_3 are scheduled on processor three, as shown in Fig. 3.

The proposed algorithm

Input: DAG's computation cost and communication cost
Output: the best solution
Initialize the parameters (Pops, Maximum_age, W_0, Maximum_iter, BRr, LOB, UPB)
Generate the population By using y_i^j= LOB + (UPB-LOB) * P(0,1)
Convert the population by using Algorithm 2
Calculate the objective function by using Algorithm 1
Calculate the fitness function
Put STj=Aj =0 j= 1,2,…., Pops
t=1
While (t<= Maximum_iter)
For j=1: Pops
iscorona($y^j(u+1)$)= false

Volume: 14 Issue: 06 Pages: 5686 - 5695 (2023) ISSN: 0975-0290

For i=1: N
If (ran < 1/3 * BRr)
Use equations Eq. 4 and Eq. 5
iscorona($y^j(u+1)$)= true
Else if (ran < 2/3 * BRr)
Use equations Eq. 6 and Eq. 7
Else if (ran < BRr)
Use equations Eq. 8 and Eq. 9
Else
$y_i^j(u+1) = y_i^j(u)$
End if
End for
Convert the obtained solution by using Algorithm 2
Calculate the objective function by using Algorithm 1
Calculate fitness function
If ($g\left(y^j(u+1)\right)$<= $g\left(y^j(u)\right)$
$y^j(u)= y'^j(u+1)$
Else
Aj= Aj+1
Endif
If ($g\left(y^j(u+1)\right) < \frac{g(y)^j(u+1)}{\Delta g(y)} \wedge ST_j = 0 \wedge iscorona(y^j(u+1))$
STj=Aj=1
End if
If ($g\left(y^j(u+1)\right) < \frac{g(y)^j(u+1)}{\Delta g(y)} \wedge ST_j = 1$)
STj=2
Aj=0
End if
If ((Aj >= Maximum_age) ∧ (STj==1))
$y_i^j(u+1)$= LOB + (UPB-LOB) * P(0,1)
STj=Aj=0
End if
End for
t=t+1
End while

Algorithm 1: Calculate the objective function of the task schedule using the Standard Genetic Algorithm (SGA) [9]

Input the schedule of tasks as shown in Fig. 3
Output the makespan
Ready_Time[Pr_j] = 0 where j = 1, 2, ……MM.
For i = 1 : NT
{
Take the first task Ta_i to be executed from LT and remove it.
For j = 1 : MM
{
If Ta_i is scheduled to processor Pr_j
Start_Time(Ta_i, Pr_j) = max{Ready_Time(Pr_j), Data_Arrive (Ta_i, Pr_j)}
Finish_Time(Ta_i, Pr_j) = Start_Time(Ta_i, Pr_j) + weight(Ta_i, Pr_j)
Ready_Time(Pr_j) = Finish_Time(Ta_i, Pr_j)
End If
}
}
makespan = max(Finish_Time)

Algorithm 2: The function that converts a continuous value to a discrete value

Function Convert (s)
R = random number between [1:5]
If (R == 1)
Use the SPV rule to convert a continuous value
Else if (R == 2)
Use the LPV rule to convert a continuous value
Else if (R == 3)
Use the round nearest function to convert a continuous value
Else if (R == 4)
Use the floor nearest function to convert a continuous value
Else
Use ceil nearest function to convert a continuous value

End if
End Function

As shown above, Algorithm 1 is used to compute the makespan by taking the schedule after converting it by Algorithm 2, which detects the method used to convert the continuous value to a discrete value by generating a random number. The proposed algorithm uses the operation of the Coronavirus Herd Immunity Optimization Algorithm to find the best solution for the makespan. We can see that speedup, efficiency, and throughput value depend on the makespan, and the more minor the makespan, the higher the speedup, efficiency, and throughput.

7. EVALUATION OF ECHIOA

We show the performance of ECHIOA by applying it to two cases. The first case is of 10 tasks and three heterogeneous processors. The second case consists of 10 tasks and three heterogeneous processors. ECHIOA was implemented as a system by MATLAB 2016. We set the initial values of the parameters Pops = 100, Maximum_age = 100, W_0 = 1, Maximum_iter = 100, LOB = 1, UPB = 3, and BRr = 0.05.

Speedup is the ratio between the results obtained by assigning all tasks to a virtual machine that gives the minimum schedule length and the results obtained by executing tasks in parallel [13].

$$\text{Speedup} = \min_{pr_j} \left(\sum_{Ta_i} \frac{weight_{i,j}}{Makespan} \right) \quad (12)$$

Efficiency is the ratio between the obtained speedup results and the total number of virtual machines used [13].

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

Throughput: The value of the throughput metric can be defined as the number of tasks executed per unit of time [14].

$$\text{Throughput} = \frac{NT.}{Makespan} \quad (14)$$

Volume: 14 Issue: 06 Pages: 5686 - 5695 (2023) ISSN: 0975-0290

7.1 Case 1

We consider a case of 10 tasks {Ta_1, Ta_2, Ta_3, Ta_4, Ta_5, Ta_6, Ta_7, Ta_8, Ta_9, Ta_{10}} to be executed on three heterogeneous processors {Pr_1, Pr_2, Pr_3}. The cost of executing every task on different processors is shown in [13]. Tab. 1 represents each task's start and finish times on different processors and the schedule obtained by ECHIOA and other algorithms. The results obtained by ECHIOA are compared with those obtained by the new genetic algorithm (N-GA) [15] and proposed particle swarm optimization (PPSO) [16]. The proposed task priority of ECHIOA {Ta_1, Ta_4, Ta_3, Ta_2, Ta_6, Ta_7, Ta_5, Ta_9, Ta_8, Ta_{10}}, task priority of N-GA {Ta_1, Ta_4, Ta_2, Ta_3, Ta_7, Ta_5, Ta_6, Ta_9, Ta_8, Ta_{10}}, task priority of PPSO {Ta_1, Ta_2, Ta_3, Ta_4, Ta_5, Ta_6, Ta_7, Ta_8, Ta_9, Ta_{10}}. Fig. 4, Fig. 5, Fig. 6, and Fig. 7 represent the results obtained by ECHIOA, N-GA, and PPSO in terms of makespan, speedup, efficiency, and throughput.

Table 1: Schedule obtained by ECHIOA and other algorithms for case 1

	N-GA			PPSO			ECHIOA		
	Pr_1	Pr_2	Pr_3	Pr_1	Pr_2	Pr_3	Pr_1	Pr_2	Pr_3
Ta_1			0-4	0-7			0-7		
Ta_2			32-50	7-14			30-37		
Ta_3	43-60			14-31			13-30		
Ta_4			4-32	31-37			7-13		
Ta_5		61-78			32-49			48-65	
Ta_6			62-89	37-72					33-60
Ta_7	60-66			72-78			37-43		
Ta_8	113-133			84-104				79-119	
Ta_9			97-113		109-142				74-90
Ta_{10}		163-175			142-154			123-135	

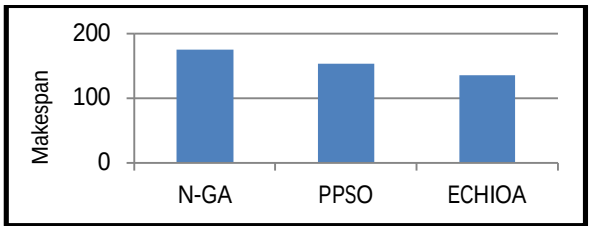


Figure 4: Comparison of makespan for case 1

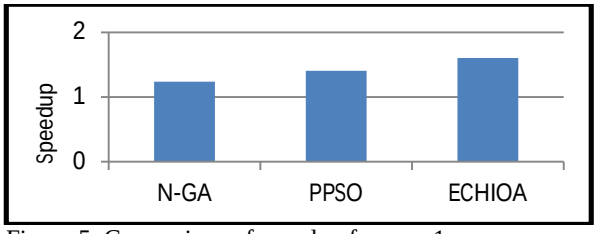


Figure 5: Comparison of speedup for case 1

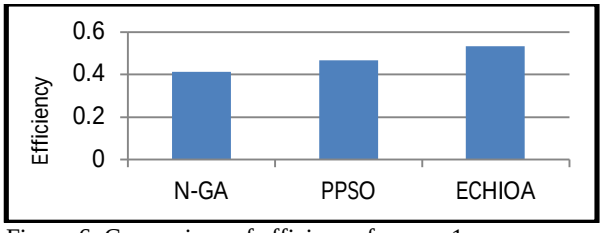


Figure 6: Comparison of efficiency for case 1

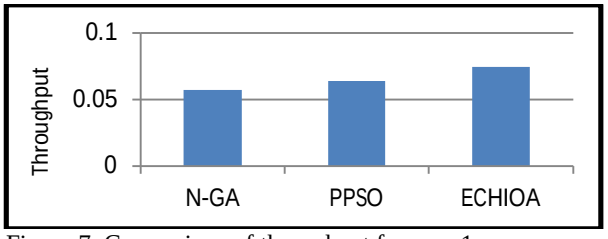


Figure 7: Comparison of throughput for case 1

7.2 Case 2

In this case, the tasks {Ta_1, Ta_2, Ta_3, Ta_4, Ta_5, Ta_6, Ta_7, Ta_8, Ta_9, Ta_{10}} are executed on three heterogeneous processors {Pr_1, Pr_2, Pr_3}. The cost of executing every task on different processors is shown in [9]. Tab. 2 represents each task's start and finish time on different processors and the schedule obtained by ECHIOA and other algorithms. The results obtained by ECHIOA are compared with the whale optimization algorithm (WOA) [17], gravitational search algorithm (GSA) [18], enhanced genetic algorithm for task scheduling (EGA-TS) [19], genetic algorithm (GA) [20], and hybrid heuristic and genetic (HHG) [21]. The proposed task priority of ECHIOA { Ta_1, Ta_6, Ta_4, Ta_5, Ta_2, Ta_3, Ta_8, Ta_9, Ta_7, Ta_{10}}, task priority of WOA {Ta_1, Ta_3, Ta_5, Ta_2, Ta_4, Ta_6, Ta_7, Ta_8, Ta_9, Ta_{10}}, task priority of EGA-TS {Ta_1, Ta_3, Ta_5, Ta_2, Ta_4, Ta_6, Ta_7, Ta_8, Ta_9, Ta_{10}}, task priority of GSA {Ta_1, Ta_3, Ta_2, Ta_6, Ta_4, Ta_5, Ta_7, Ta_8, Ta_9, Ta_{10}}, task priority of GA {Ta_1, Ta_2, Ta_4, Ta_5, Ta_9, Ta_3, Ta_7, Ta_6, Ta_8, Ta_{10}}, task priority of HHG {Ta_1, Ta_2, Ta_6, Ta_3, Ta_4, Ta_5, Ta_8, Ta_7, Ta_9, Ta_{10}}. Fig. 8, Fig. 9, Fig. 10, and Fig .11 represent the results obtained by ECHIOA, WOA, EGA-TS, GSA, GA, and

Volume: 14 Issue: 06 Pages: 5686 - 5695 (2023) ISSN: 0975-0290

HHG in terms of makespan, speedup, efficiency, and throughput.

Table 2: Schedule obtained by ECHIOA and other algorithms for case 2

	WOA			EGA-TS			GSA			GA			HHG			ECHIOA		
	Pr1	Pr2	Pr3	Pr1	Pr2	Pr3	Pr1	Pr2	Pr3	Pr1	Pr2	Pr3	Pr1	Pr2	Pr3	Pr1	Pr2	Pr3
Ta1		0-21			0-21			0-21		0-22				0-21			0-21	
Ta2			38-56	38–60					38-58	22-44				21-39			21-39	
Ta3		21-48			21-48			21–48		44-76				39-66			39-66	
Ta4		48-58			48-58		50-57				51-61				50-54	64-71		
Ta5	34-63					34-69			56-91			35-70			54-89			34-69
Ta6		58-75			58-75			48-65			61-78		38-64			38-64		
Ta7	64-78				75-100		64-78			76-90				66-91			66-91	
Ta8		75-98		80-109				68-91			78-101		65-94			71-100		
Ta9	86-10					90-98			91-99			74-82			89-97			78-86
Ta10	1	108-124		109-122			106-122			101-117		104-117			100-113			

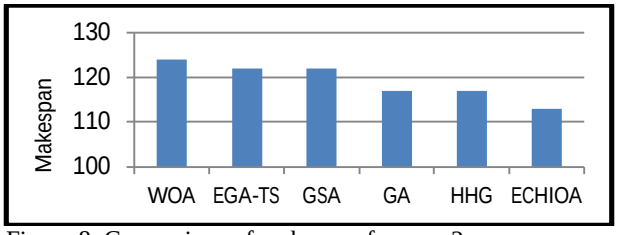


Figure 8: Comparison of makespan for case 2

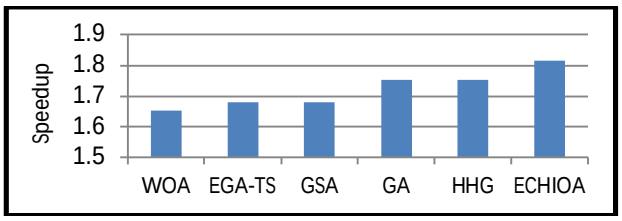


Figure 9: Comparison of speedup for case 2

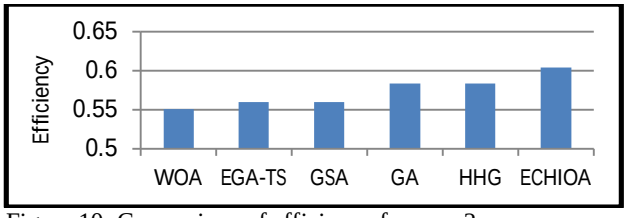


Figure 10: Comparison of efficiency for case 2

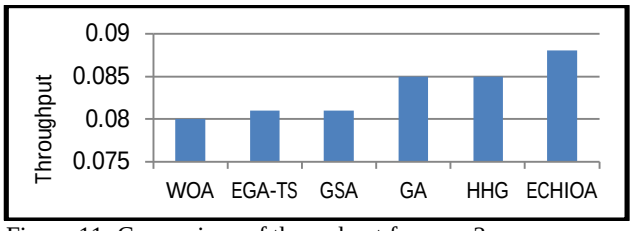


Figure 11: Comparison of throughput for case 2

7.3 Case 3

To analyze the behavior of the ECHIOA, we consider five cases with three heterogeneous processors. The tasks are 20, 30, 50, 70, and 100. The communication cost is equal to one, and the computation cost of each task on different processors is randomly generated from 1 to 100, respectively. The results obtained by ECHIOA are compared with the Efficient Cooperation Search Algorithm (ECSA) [22]. We ran our proposed algorithm and ECSA one more time, and the results obtained by the proposed algorithm are shown in Tab. 3, and the results obtained by ECSA are shown in Tab. 4. Fig. 12, Fig. 13, Fig. 14, and Fig .15 represent the results obtained by

Volume: 14 Issue: 06 Pages: 5686 - 5695 (2023) ISSN: 0975-0290

ECHIOA and ECSA in terms of makespan, speedup, efficiency, and throughput.

Table 3: The results obtained by ECHIOA for case 3

number of tasks	makespan	speedup	efficiency	Throughput
20	451	1.90	0.63	0.04
30	636	1.95	0.65	0.04
50	1309	1.81	0.60	0.03
70	1967	1.77	0.59	0.03
100	2854	1.76	0.58	0.03

Table 4: The results obtained by ECSA for case 3

number of tasks	makespan	Speedup	Efficiency	Throughput
20	380	2.26	0.75	0.05
30	618	2.0	0.66	0.04
50	1316	1.80	0.60	0.03
70	2127	1.63	0.54	0.03
100	2916	1.73	0.57	0.03

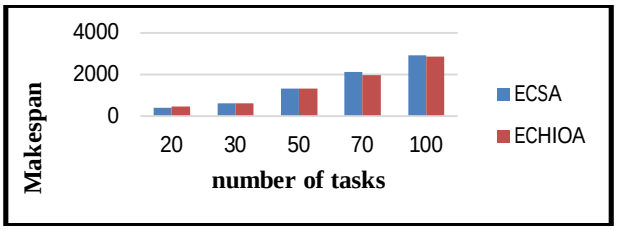


Figure 12: Comparison of makespan for case 3

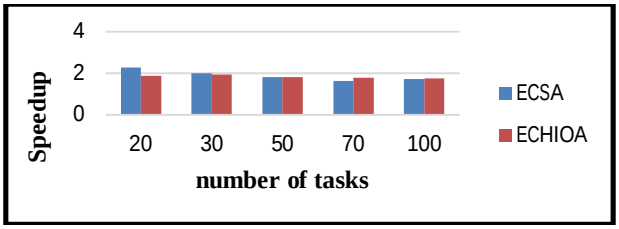


Figure 13: Comparison of speedup for case 3

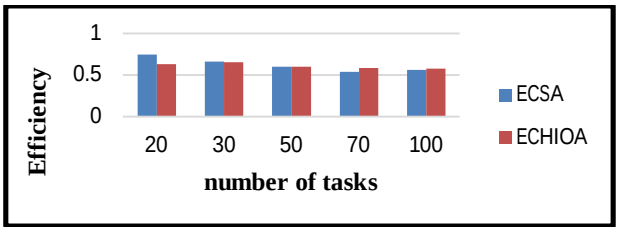


Figure 14: Comparison of efficiency for case 3

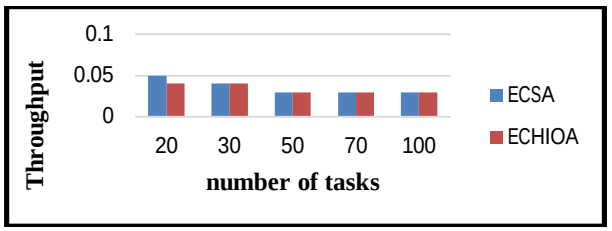


Figure 15: Comparison of throughput for case 3

7.3 Discussion

According to the results in Fig. 4, Fig. 5, Fig. 6, and Fig. 7, it is found that the makespan of ECHIOA is reduced by 22.8% and 12.3% of NGA and PPSO, respectively. The speedup of ECHIOA is improved by (29.6%) and (14.1%) for NGA and PPSO, respectively. The efficiency of ECHIOA is improved by (29.6%) and (14.1%) for NGA and PPSO, respectively. The throughput of ECHIOA is improved by (29.8%) and (15.6%) of NGA and PPSO, respectively. According to the results in Fig. 8, Fig. 9, Fig. 10, and Fig. 11, it is found that the makespan of ECHIOA is reduced by 8.8%, 7.3%, 7.3%, 3.4%, and 3.4% about WOA, EGA-TS, GSA, GA, and HHG, respectively. The speedup of ECHIOA is improved by (9.7%), (7.9%), (7.9%), (3.5%), and (3.5%) of WOA, EGA-TS, GSA, GA, and HHG, respectively. The efficiency of ECHIOA is improved by (9.6%), (7.8%), (7.8%), (3.4%), and (3.4%) of WOA, EGA-TS, GSA, GA, and HHG, respectively. The throughput of ECHIOA is improved by (10%), (8.6%), (8.6%), (3.5%), and (3.5%) of WOA, EGA-TS, GSA, GA, and HHG, respectively. According to the results in Fig. 12, Fig. 13, Fig. 14, and Fig. 15, when the number of tasks is increased, the ECHIOA outperformed the ECSA regarding makespan, speedup, efficiency, and throughput.

8. Conclusion and Future Work

In order to get near-optimal results for the problem of scheduling tasks in the cloud computing environment, efficient strategies for the optimal mapping of the tasks are required. This paper has presented a new efficient approach based on the coronavirus herd immunity optimizer algorithm called the efficient coronavirus herd immunity optimizer algorithm (ECHIOA) to solve the scheduling task problem in the cloud computing environment. The system has comprised a small number of fully interconnected heterogeneous processors. The algorithms have been compared against the algorithms according to makespan, speedup, efficiency, and throughput. The comparative analysis explained that the proposed algorithm performs better in all cases. In our future work, we will develop an efficient cuckoo search algorithm for optimizing scheduling tasks in a cloud computing environment to minimize the makespan and cost and maximize the speedup, efficiency, throughput, and resource utilization.

References

[1] X. Chen, L. Cheng, C. Liu, Q. Liu, J. Liu *et al.*, A woa-based optimization approach for task scheduling in cloud computing systems,*IEEE Systems Journal, 14*(3), 2020, 3117–3128.

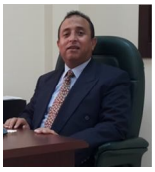
[2] I. Attiya, M. Abd Elaziz and S. Xiong, Job scheduling in cloud computing using a modified harris hawks optimization and simulated annealing algorithm,*Computational Intelligence and Neuroscience, 2020*(1), 2020, 1-17.

[3] G. Natesan and A. Chokkalingam, An improved grey wolf optimization algorithm based task scheduling in cloud computing environment, *The International Arab Journal of Information Technology, 17*(1),

Volume: 14 Issue: 06 Pages: 5686 - 5695 (2023) ISSN: 0975-0290

2020, 73-81.
[4] S.M.G. Kashikolaei, A.A.R. Hosseinabadi, B. Saemi, M.B. Shareh, A.K. Sangaiah *et al.*, An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm,*Journal of Supercomputing, 76*(8), 2020, 6302–6329.
[5] A. Alameen and A. Gupta, Fitness rate-based rider optimization enabled for optimal task scheduling in cloud,*Information Security Journal, 29*(6), 2020, 310–326.
[6] KR Prasanna Kumar and K. Kousalya, Amelioration of task scheduling in cloud computing using crow search algorithm,*Neural Computing and Applications, 32*(10), 2020, 5901–5907.
[7] L. Abualigah and A. Diabat, A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments,*Cluster Computing, 24*(1), 2021, 205–223.
[8] M. Gokuldhev, G. Singaravel and N.R. Ram Mohan, Multi-objective local pollination-based gray wolf optimizer for task scheduling heterogeneous cloud environment,*Journal of Circuits, Systems and Computers, 29*(7), 2020, 1–24.
[9] A. Younes, A. BenSalah, T. Farag, F. A.Alghamdi and U. A. Badawi, Task scheduling algorithm for heterogeneous multi processing computing systems,*Journal of Theoretical and Applied Information Technology, 97*(12), 2019, 3477-3487.
[10] M. A. Al-Betar, Z. A. A. Alyasseri, M. A. Awadallah and L. A. Doush, Coronavirus herd immunity optimizer (CHIO),*Neural Computing and Applications, 33*(10), 2021, 5011–5042.
[11] I. Dubey and M. Gupta, Uniform mutation and SPV rule based optimized PSO algorithm for TSP problem, in *Proc. of the 4th Int. Conf. on Electronics and Communication Systems, Coimbatore, India,* 2017, 168–172.
[12] L. Wang, Q. Pan and F. M. Tasgetiren, A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem,*Computers & Industrial Engineering, 61*(1),2011, 76-83.
[13] A. Mishra, M. N. Sahoo and A. Satpathy, H3CSA: A makespan aware task scheduling technique for cloud environments,*Transactions on Emerging Telecommunications Technologies*,*32*(10), 2021, 1-20.
[14] S. Nabi, M. Ibrahim and J. M. Jimenez, DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing,*IEEE Access, 9*(1), 2020, 61283-61297.
[15] B. Keshanchi, A. Souri and N. Navimipour, An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing, *Journal of Systems and Software, 124*(1), 2017, 1-21.
[16] T. Biswas, P. Kuila and A.K. Ray, A novel workflow scheduling with multi-criteria using particle swarm optimization for heterogeneous computing systems,*Cluster Computing, 23*(4), 2020, 3255–3271.
[17] S. R. Thennarasu, M. Selvam and K. Srihari, A new whale optimizer for workflow scheduling in cloud computing environment,*Journal of Ambient Intelligence Humanized Computing, 12*(3), 2020,3807-3814.
[18] T. Biswas, P. Kuila, A. K. Ray and M. Sarkar, Gravitational search algorithm based novel workflow scheduling for heterogeneous computing systems,*Simulation Modelling Practice and Theory, 96*(1), 2019, 1-21.
[19] M. Akbari, H. Rashidi and SH Alizadeh, An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems,*Engineering Applications of Artificial Intelligence, 61*(3), 2017, 35–46.
[20] A. Y. Hamed and M. H. Alkinani, Task scheduling optimization in cloud computing based on genetic algorithms,*Computers, Materials & Continua, 69*(3), 2021, 3289-3301.
[21] M. Sulaiman, Z. Halim, M. Lebbah, M. Waqas and S. Tu, An evolutionary computing-based efficient hybrid task scheduling approach for heterogeneous computing environment,*Journal of Grid Computing, 19*(1), 2021, 1-31.
[22] A.Y. Hamed, M. K. Elnahary, F. S. Alsubaei and H. H. El-Sayed, Optimization Task Scheduling Using Cooperation Search Algorithm for Heterogeneous Cloud Computing Systems,*Computers, Materials & Continua, 74*(1), 2023, 2133-2148.

Biographies and Photographs



A. Younes received his PhD degree in Sept. 1996 from South Valley University, Egypt. His research interests include Artificial Intelligence and genetic algorithms; specifically, in the area of computer networks. Recently, he has started conducting a research in the area of Image Processing. Currently, he works as an Professor SohagUniversity, Egypt. Younes always publishes the outcome of his research in international journals and conferences.

M. Kh. Elnahary Received the B.S degree from computer science department, Sohag University, Egypt. His interests in task scheduling and computer networks.

Hamdy H. El-Sayed Received the PhD degree in wireless ad hoc network routing protocols from computer science department sohag university Egypt march ,2015. His research interests are in the areas of ad hoc routing protocols and sensor networks, Internet of Things, cloud computing and security.