JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2920 - 2931
https://publishoa.com
ISSN: 1309-3452

Comparative Performance Analysis of Task Scheduling In Cloud Computing Using ACO, PSO, Firefly And Loa Algorithms

[1]N. Karunya, [2]Dr. T. Deepa
[1]Research Scholar, Department of Computer Science, Sri Ramakrishna College of Arts and Science for Women, Coimbatore, Tamilnadu, India.
[1]Assistant Professor , Sri Ramakrishna College of Arts & Science, Coimbatore, Tamilnadu, India.
[2] Assistant Professor, Department of Computer Science, Sri Ramakrishna College of Arts and Science for Women, Coimbatore, Tamilnadu, India.

ABSTRACT

Cloud computing is a client-driven requirement that provides many resources with the goal of sharing them as a service over the internet. It is a stage in which Cloud clients can access a unique pool of resources and virtualization. Cloud computing provides dynamic resource allocation on demand, as well as scalability, availability, and various services. The cloud provides services to organizations such as processing, storage, server, and applications that are located in remote areas by utilising cloud servers. The system's performance suffers if tasks are not properly scheduled As a result, resource allocation is critical to improving overall system performance. The scheduling of user tasks is critical for improving the performance of cloud services. Cloud providers must effectively schedule their resources for maximum utilisation and user satisfaction.In this paper has given a optimize the scheduling algorithm based on Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO),Firefly Algorithm, Lion Optimization Algorithm (LOA) to improve the performance of cloud task scheduling. The simulation results show the effectiveness of the comparison by using research parameters like Cost, Resource Utilization, Make span, Execution Time and Completion Time.

Keywords: Cloud Computing, Task Scheduling, Ant Colony Optimization, (ACO), Particle Swarm Optimization (PSO), Firefly Algorithm, Lion Optimization Algorithm, Virtual Machine

I. INTRODUCTION

Cloud computing is becoming a necessity in the IT industry due to the wide range of services it provides. The Internet is used to deliver Cloud services. Devices that want to use the Cloud's services should be able to access the internet. Devices must have minimal memory, as well as a lightweight operating system and browser. Cloud computing has many advantages: it saves money even though there is no need to install a large amount of resources at first; it gives scalability and flexibility, enabling users to increase or decrease the number of services as needed; and it has a low cost because all resources are managed by the Cloud providers. Cloud computing is the result of combining standard computer methods with network technologies such as parallel computing, grid computing, distributed computing, load balancing,utility computing, network storage, virtualization, etc[1]. The primary goal is to schedule tasks to Virtual Machines (VMs) based on adaptable time, which entails determining a regular pattern in which tasks can be executed while adhering to transaction logic restrictions. [2].

Cloud computing had also recent times gotten a lot of attention as just a hopeful way of delivering information and communication technology (ICT) services as a utility[3]. It is critical to enhance the utilization of data centre resources that are operating in the most dynamic workload environments inside the tool for providing these services. Datacenters are critical components of cloud computing. Hundreds of thousands of virtual servers typically run in a single datacenter at any given time, hosting numerous tasks, whereas the public cloud continues to receive batches of task requests. Throughout this context, one must notice a few target servers among a large number of powered on servers that can complete a batch of arriving tasks. As a result, task scheduling is an important issue that has a significant impact on the performance of cloud service providers. Conventional optimization strategies are deterministic, fast, and produce exact outcomes, but they frequently become trapped on local optimization. The task scheduling problem has a Non Polynomial -complete complexity, which means it has an extremely large search area with a correspondingly large number of potential solutions and takes much longer to find the optimal solution. There really is no ready-made and well-defined procedure for resolving problems in these kind of circumstances. In the cloud, however, it's indeed acceptable to discover a close solution, ideally in a short amount of time. IT practitioners are focusing on heuristic methods in this framework. In cloud computing, there are numerous scheduling algorithms. The scheduling algorithm's main advantage has been its high performance. , Round-Robin, FCFS,Max-Min algorithm, Min-Min algorithm, and meta-heuristic algorithms (PSO, GA,ACO, Simulated annealing, Tabu search, and many more) are most common scheduling algorithms [4]. Simulated annealing, , and ant colony optimization, as well as particle swarm intelligence optimization algorithms,genetic

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2920 - 2931
https://publishoa.com
ISSN: 1309-3452

algorithms are excellent for solving NP problems. Cloud computing task scheduling problems have been solved using genetic algorithms, particle swarm optimization algorithms, and ant colony algorithms.
A task scheduling algorithm is a technique for matching or allocating tasks to data centers. In general, no absolutely perfect scheduling algorithm exists due to competing scheduling objectives. A good scheduler enforces an appropriate negotiated settlement or includes a mixture of scheduling algorithms based on the application [5]. Depending on the algorithm used, a problem can be solved in seconds, hours, or even years. The amount of time it takes to execute an algorithm is used to determine its efficiency. The time complexity function relating the input is used to express the execution time of an algorithm. The literature contains several types of time complexity algorithms. If a polynomial time algorithm is used to solve a problem, The trouble is manageable, workable, effective, or quick enough to be solved on a computational machine. A set of problems could be handled as a largely divided in computational complexity theory focused on a particular resource.

- Class NP is a collection of decision feasible solutions in polynomial time on a nondeterministic Turing machine, but just a potential solution of a Class NP problem can be affirmed by a polynomial time algorithm, implying that the issue can be validated fast.
- Class NP-complete [6] is the collection of decision problems whereby all other NP problems can be linear converted, and an NP-complete problem must be in class NP. NP-complete troubles are usually more difficult than NP problems.
- Class NP-hard is a set of optimization algorithms to which all NP troubles can be polynomially transformable, but rather an NP-hard problem isn't always in class NP.
- The problem of matching tasks to different sets of resources is expressed directly as a triple (T, S, O), where 'T' is the set of tasks, each of which is an instance of the problem, 'S' is the set of feasible solutions, and 'O' is the problem's objective.
- Scheduling troubles are also classified into two types based on objective O: optimization problems and decision problems. An optimization problem necessitates the selection of the best solution from among all possible solutions in set S. In contrast to optimization, the goal of a decision problem is relatively simple. The problem requires a positive or negative answer to whether the objective is met for a given feasible solution s € S. Clearly, optimization problem is harder than decision problem.

II. OPTIMIZATION ALGORITHMS

2.1. Ant Colony Optimization Algorithm

Colorni first proposed the ACO algorithm in 1991, and Dorigo proposed the first ant system (AS) in his Ph.D. thesis in 1992. The ACO is a meta-heuristic algorithm that uses pheromone information to find the shortest path from a food source to the nest without using visual cues. When ant colonies search for food, they leave chemical components known as pheromones on their trails. The pheromone left on the ground increases as the ants walk through the path. So the next ant can choose one route with a probability proportional to the quantity of pheromone, this positive responses system will likely lead to the formation of an obvious approach from their nest to the food source. The following are the main characteristics of real ants: (1) True ants prefer the pheromone trail with the highest intensity. (2) The more pheromone is left on a path, the shorter the distance. (3) True ants interact indirectly through pheromones. Above behaviour of identity real ants in search of the shortest path aided in the development of the ACO. Artificial ants collaborate to find a solution by exchanging information via pheromone deposits on paths.

ACO techniques is used to solve single - objective optimization problems that require finding pathways to objectives. It's been used to solve problems like the travelling salesman problem, multidimensional knapsack problem, job shop scheduling, quadratic assignment problem, grid and cloud task scheduling, and others. The first step in using ACO to solve any problem is to map an ant system to the problem at hand. The number of ants used for scheduling independent tasks in grid [5] or cloud is below or equal to the total number of tasks. Each ant begins with an arbitrary task ti and a resource Rj for completing this task. The following probable function is used to calculate the task to be executed and the resource on which it is performed:

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\Sigma(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}$$

Where

τ_{ij} denotes the pheromone value related to task t_i and resource r_j
η_{ij} denotes the heuristic function
α determines the influence of pheromone value
β determines the influence of heuristic function

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2920 - 2931
https://publishoa.com
ISSN: 1309-3452

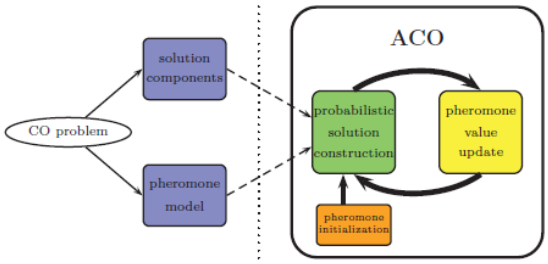

Figure 1: ACO Framework

Figure 1 graphically depicts the basic operation of an ACO algorithm. Provided a CO problem to solve, one must first derive a finite set C of system elements which are used to assemble CO problem solutions. Second, a set of pheromone values T must be defined. From a technical standpoint, this set of values is known as the pheromone model, which is a parameter way of estimating. One of the most important aspects of ACO is the pheromone model. I T pheromone values are typically associated with solution components. The pheromone model is used to generate solutions to problems probabilistically by assembling them from a set of solution components. The ACO approach, in general, tries to solve an optimization problem by repeating the two steps below. [6]:

1. candidate solutions are built using a pheromone model, which is a parameterized probability distribution over the solution space; 2. candidate solutions are being used to adjust pheromone values in a way that is thought to bias destiny sampling toward high-quality solutions. The pheromone update aims to focus the search on areas of the search space with high-quality solutions. It makes this same insinuation that ensures successful are made up of good system elements.
2. The algorithmic steps of the ACO designed by Dorigo and Gambardella in 1997are outlined as the following.

Algorithm:
1. Initialize
Set initial pheromone on each edge
2. Loop/ at this level each loop is called an iteration */*
Each ant is positioned on a starting node
a. Loop/ at this level each loop is called a step */*
Each ant applies a state transition rule to incrementally build a solution and a local pheromone updating rule
Until all ants have built complete solutions
b. A global pheromone updating rule is applied
Until stopping criterion
1. Output the global best tour.

2.2. Particle Swarm Optimization

Particle swarm optimization is a modern, non-traditional optimization algorithm. It is population-based, and it is inspired by natural animal or insect behaviour, such as bird flocking and schools of fish. Dr. Kennedy and Dr. Eberhart first tried to introduce it in 1995. [6]. PSO has been designed to solve non-linear optimization problems, and it's now used for a range of real applications. It is a discrete problem to schedule cloud computing tasks. The particle swarm optimization algorithm, on the other hand, is best suited for solving continuous optimization problems. The particle swarm optimization (PSO) algorithm is implemented in a discrete version. [7], Each particle's position vector is assigned a value of 0 or 1. Particle velocity does not anymore refer to the speed of particles in continuous space flight. It denotes the likelihood of calculating particle position vector values to 0 or 1. The position and velocity of each particle should be redefined based on the cloud computing task scheduling characteristics [8]. The position of each particle in the particle swarm represents the task scheduling scheme. The position of the first I particles can be represented as the expression

$$x_{ij} \in \{0,1\}, \sum_{i=1}^{m} x_{ij} = 1, i \in \{1,2,\cdots,m\}, j \in \{1,2,\cdots,n\}$$

using the distribution relationship matrix.

$$v_{ij} \in [-v_{max}, v_{max}], i \in \{1,2,\cdots,m\}, j \in \{1,2,\cdots,n\}$$

In the discrete particle swarm optimization algorithm, the fuzzy function is introduced to ensure the value of the position vector is 0 or 1.

$$sig(v_{ij}^{k+1}) = \frac{1}{1+exp(-v_{ij}^{k+1})}.$$

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2920 - 2931
https://publishoa.com
ISSN: 1309-3452

The algorithm is started with a random 'n' population, a search space, and a random direction to move. In PSO, each particle is referred to as a'swarm,' and it moves through the search space, adjusting its position based on its own best position or the best position of its neighbours, in order to find the best solution.

Algorithm:

1. Initialize the swarm (n) form the solution space
2. Evaluate the fitness value of each particle. (Objective function)
3. Evaluate individual (Pbest) and global bests (Gbest).
4. Calculate the velocity of each individual particle.

$$Vj = Vj(i-1) + c1r1\,[pbest,j - X\,j(i-1)] + c2r2\,[gbest,j - X\,j(i-1)] + \ldots.(1)$$

Where

c1, c2= positive acceleration constants.
r1, r2 = random numbers.

5. Calculate the position of each particle.

$$X\,j(i) = X\,j(i-1) + Vj(i) \ldots.(2)$$

Go to step2, and repeat until termination condition

2.3. Firefly algorithm:

The inspiration for this firefly algorithm came from the swarm behaviour of fireflies. Fireflies are known to exist in groups and to behave in a swarm-like manner. The firefly's blinking light is an attractive feature that they use to attract mates and defend themselves from other predators. A swarm of fireflies will usually follow the brightest one. All of the other fireflies with lower light intensities migrate toward those with higher light intensities. As a result, as the distance between the fireflies grows, so does the intensity of the light [8].

The primary function of a firefly's flash is to act as a signal to attract other fireflies. Fireflies use their flashing signal to attract mates and prey, as well as to share food with others. The firefly algorithm, like other met heuristics optimization methods, generates a random initial population of viable candidate solutions. In the solution search space, all fireflies in the population are managed with the goal of collectively sharing knowledge among fireflies to guide the search to the best location in the search space. Each particle in the population is a firefly that moves in the multidimensional search space with an attractiveness that really is constantly updated based on the stages of the proposed algorithm, that will be described in greater detail in the next section. We can now idealise some of the flashing characteristics of fireflies in order to create firefly-inspired algorithms. Use the following three idealised rules to describe our new Firefly Algorithm (FA) for simplicity:

(1) Because all fireflies are unisex, one firefly will be attracted to another firefly regardless of gender;

(2) Desirability seems to be proportional to brightness, so if two flashing fireflies are attracted to each other, some less brighter one will move towards the brighter one. The desirability is simply breathtaking, and both decrease as the distance between them grows. If there is no brighter one than a particular firefly, it will move at random.

(3) The landscape of an objective function influences or determines the brightness of a firefly. In the case of a maximization problem, the brightness can simply be proportional to the objective function's value. Other types of brightness can be defined in a manner similar to how the fitness function in genetic algorithms is defined [10]. The basic steps of the Firefly Algorithm (FA) can be summarized as pseudo code using these three rules, which are shown below.

Firefly Algorithm:

Objective function f(x), x(x1,…,xd)T Generate initial population of fireflies xi (i = 1,2,…,n) Light intensity Ii at xi si determined by f(xi) Define light absorption coefficient
while (t< Max_Generation)
fori = 1: n all n fireflies
forj = 1: i all n fireflies
if (Ij>Ii), move firefly i towards j in d-dimension;
end if
Attractiveness varies with distance r via exp [-r]
Evaluate solutions and update light intensity
end for j
end for i
Rank the fireflies and find the current best
end while
Postprocess results and visualization

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2920 - 2931
https://publishoa.com
ISSN: 1309-3452

Objective Function : The objective function, also known as the fitness function, is the function that defines and formulates the conditions necessary for achieving the optimization goal. The improvement in this given problem should result in a reduction in the total execution time of all tasks in the cloud environment, resulting in improved performance. As a result, the goal role strives to reduce total execution of all sub - tasks. Execution time of a task depends upon two variable entities namely instruction execution speed as powered by the processor core of the VM and size or length of the instructions of each task. Each resource or VM has its own execution speed given in units of Million Instructions per second (MIPS) which is denoted in vector R_i where $i \in [1, n]$Each subtask has its length is represented in units of Million Instructions encoded in Li where I ∈ [1, l]. Each task's execution time in seconds is calculated by dividing the task's length by the speed of the VM assigned to it. Each firefly's execution time or fitness function is then provided by:

$$fi = \sum_{i=1`}^{i} \frac{Li}{Ri} \quad -- \quad (1)$$

$$Fi = \sum_{i=1`}^{i} \frac{1}{fi} \quad -- \quad (2)$$

The algorithm's goal is to minimise the total execution time of all tasks obtained in (1). The inverse relationship of execution time is given by Eq. (2). The longer the execution time of tasks fi obtained by a specific firefly I the lower the value obtained for F for that firefly. Hence the best solution is to find a firefly or chromosome with maximum fitness F, i.e.

Most fit firefly = $\max_i F_i$

The value of fluorescein in the basic firefly algorithm is to ascertain if the firefly's place is the best location, and when it is, you can guide more fireflies to their own close to find the function of the mouth. Using the improved value of the updated fluorescence can help to avoid going to fall into the neighbourhood convergence, that can help to search; this improvement is much more in line with the multi-tasking assets in the reasonable use of cloud computing. There really is no doubt that the completion time between tasks on the cloud server will be reasonably close to the processing time, ensuring that the objective function is kept to a minimum. The improved firefly algorithm is used in cloud computing. All processing tasks are carried out using string encoding, the customer tasks are randomly selected to be processed in the cloud, all tasks are numbered, each firefly represents a solution, and the location of fireflies is determined. The length represents the total number of operations, and the number of fireflies represents the total number and space of a way to solve. The minimum value of the cloud computing task completion time is the value of Firefly's goal function.

Light intensity and attractiveness: The firefly algorithm's execution is dependent on the discrepancy in intensity of light and the contextualisation of attractiveness; light intensity symbolises a firefly's brightness and decreases with range from its source. Moreover, because of light absorption by air, attractiveness varies with the extent of absorption (Equation 3).

I = I (4) o exp-$^{\gamma r2}$ --(3)

Where Io includes the initial light intensity and 0 reflects the absorption coefficient.

This same higher the intensity of the light, the good the command of requests inside the request queue. A firefly's desirability is limited by it's own brightness, which is affiliated with the ciphered fitness values. The original FA defines the attractiveness function β_r for any flat decreasing function as follows

$$\beta(r) = \beta_0 \exp -\gamma r^2 \quad -- \quad (4)$$

where β (r) is the attractiveness of a firefly at a distance r, r being the distance between two fireflies. 0 represents the brightness of a brighter firefly, and represents a constant light absorption coefficient. For any two fireflies, firefly I and another brighter firefly j, we first calculate the distance (r) between firefly I and firefly j using Equation 3. We then use Equation 4 to calculate the attractiveness of firefly j observed by firefly I at distance r. If the attractiveness of firefly j exceeds the brightness (light intensity) of firefly I firefly I will move toward firefly j; otherwise, firefly I will start moving at random.

The brightness and attractiveness of firefly: The locations of the fireflies must be considered when comparing the brightness of any two fireflies. The attractiveness and brightness of two fireflies decrease dramatically as the distance between them grows. Furthermore, if a firefly does not find another firefly in its vicinity, it will fly in an undetermined direction. The algorithm contrasts the attractiveness of the new and old firefly positions. If the new position has a higher attractiveness value, the firefly is moved to it; otherwise, the firefly stays in its current position. The FA's termination criterion is based on an arbitrary number of iterations or even a predefined fitness value. [11][12].

The brightest firefly moves at random according to the following equation: Each firefly has an attractiveness value, which indicates how well it can attract other fireflies in the swarm. The attractiveness will diverge with its distance factor dij at the locations Xi and Xj, between the two corresponding fireflies I and j, as shown by the comparison.

$$dij = |Xi - Xj| \quad (5)$$

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2920 - 2931
https://publishoa.com
ISSN: 1309-3452

The attractiveness function β of the firefly is computed as:

$$\beta = \beta 0\ e^{-\gamma\ r2} \quad (6)$$

where $\beta 0$ is attractiveness at r=0 and □ is coefficient of light absorption.

The movement of the less bright firefly toward the brighter firefly is computed by

$$Xi = Xi + \beta 0\ e^{-\gamma r2ij}\ (Xj - Xi) + \alpha\ (r\ and\text{-}1/2) \quad .. \ (7)$$

where α is the randomization parameter and rand is a randomly selected number in the interval [0, 1].

The movement towards attractiveness firefly: When firefly I is drawn to the brighter firefly j, its movement is defined by the equation below,

$$xi = xi + \beta 0\ x\ e^{-\gamma rij2}\ x\ (xj - xi) + \alpha x\ (rand - 1/2) \ldots (8)$$

In equation (4), xi is current position, $\beta 0\ x\ e^{-\gamma rij2}\ x\ (xj - xi)$ is the brightness of the firefly and the last term represent random motion [13]. The $\alpha(rand - 1/2)$ is a firefly's random movement. The coefficient α is a randomisation parameter determined by the problem of interest with $\alpha \in [0, 1]$, while *rand* is a random number obtained from the uniform distribution in the space [0, 1] [9, 23].

The desirability tends to vary with the range d_{ij} among fireflies I and j. Light intensity reduces as one tries to move away from the observer, and light is also soaked up with in medium. [14]. As a firefly's attractiveness is proportional to the light intensity seen by adjacent fireflies, attractiveness function (dij) of a firefly can be a monotonically decreasing function presented as,

$$\beta(d_{ij}) = \beta_0 e^{-\gamma d_{ij}} \quad (9)$$

where dij is the distance between firefly i and firefly j, γ is the light absorption coefficient, and $\beta 0$ is the attractiveness at $dij = 0$. We take $\beta 0 = 1$, $\gamma = 1$.

2.4. Lion Optimization Algorithms:

The LOA was based on lion behaviours such as chasing, mating, and protection. Lions have two hierarchical behaviours: inhabitant behaviour and migrant behaviour. Residents live in prides, which are social gatherings. An occupant lion can transform into a traveller, and vice versa. The underlying populace in the LOA is generated haphazardly across the arrangement space, with each arrangement known as a "Lion." (percent N) of the lions in the inhabitants are selected at random to be nomad lions, while the remainder are inhabitants. Residents are assigned to (P) prides at random. (percent S) of the lions in each pride are female, while the rest are male. Nomad lions, on the other hand, have the ratio overturned. The able to roam percentage is (percent R), and the breeding percentage of female lions is (percent Ma). This same immigrant rate of female lions in each pride is (percent I)[15].

The lion algorithm's process is as follows [16]:

- Begin with random populations.
- Create prides and lions.
- One lion particle equals:
  (a)Choose a random female lion for hunting (b)Each female lion chooses the best position in the pride
  (c)Select this same weakest lion pride from of the population becoming the nomad (d)Each pride evaluate the imigration rate and become the nomad
- evaluating the objective functions to pick the optimal females as well as fill the mepty positions with female lions relocated from the territory.

The completion time of the resources allocated to the virtual machines is computed by Minimum Execution Time. The task is assigned to machines that are focused on completing the task in the shortest amount of time possible while utilising essential MET facts which sometimes deal with significant weight imbalance. The job does not require access to a machine.

- like Most, the jobs are delegated to the machineries, out of which we estimated the final tally.
- A large number of jobs that are then prioritised.
- The current development id and time are used to calculate the process's execution and completion times.
- Determine the machine's performance time for the process.
- Estimate the time it will take machine m to complete task t.
- The predictable or minimum completion time is estimated using the sum of the execution time to complete the processes by the machine and the initial execution time of the current task on the current processor or machine.
- Repeat this process until all iterations have been finished and then stop.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2920 - 2931
https://publishoa.com
ISSN: 1309-3452

- Determine the machine's minimum execution time for the task until all employment are drained.

Initialize population: Our goal in this job is to fix the job scheduling problem for cloud computing while reducing the optimizer makespan, which would be the maximum execution time for any and all tasks. As a result, we must assign a lion to each underlying solution.

Parameter	Definition of the parameters
$\%N$	Percent of nomad lions
P	Number of prides
$\%S$	Percent of female lions in each pride
$\%R$	Roaming percent
$\%Ma$	Mating percent of female lions
$\%I$	Immigrate rate of female lions in each pride

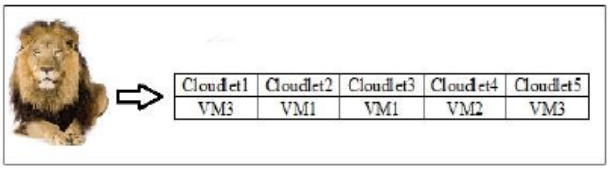

Figure 2 : A lion representation

A lion represents a task scheduling solution that really is randomly initialised by mapping cloud tasks (cloudlets) to cloud resources (virtual machines (VMs). Figure 2 depicts a lion representing a schedule of five task given to three VMs at random. As a result, each lion represents a random schedule solution, and the initial lion population is generated at random across the LOA algorithm's solution space. The suggested algorithm's task is to find this same best lion (solution) with best fitness value. The fitness value is the solution's makespan, which is the total time required to complete the tasks. Furthermore, each lion is aware of its own best solution (task schedule) and the global best solution, which would be updated incrementally during optimization.

Hunting: A few females in each pride look for a food source in a collective to provide food for their pride. Such hunters employ distinct tactics to conquer and capture their prey. When it came to hunting, lions generally followed similar patterns [17]. Figure 3 depicts Stander's classification of the lions' stalking roles, which he divided into Left Wing, Center Wing, and Right Wing positions. During hunting, each lioness adapts her claim based under her own and the roles of the other lionesses in the group. Because some of these hunt occupy prey and attack from different directions while hunting, we employ

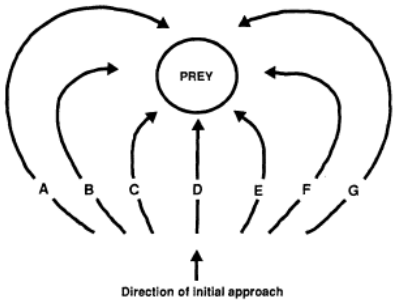

Figure 3: A schematic of generalized lions hunting behavior [18].

The hunters are selected at random and pounce on the dummy prey. After a little while, this process would be defined based on the hunting lion's group. PREYX will escape from the hunter if the hunter's ability and subtlety are improved over time, and PreyX's new role is as continues to follow:

$$\text{Prey}_y = \text{PreyX}+\text{random}(0,1)\times\text{Pi} \times (PreyX - Hunter)$$

Where PreyX denotes the prey's current position, hunter attacks the prey in the new position, and pi denotes the hunter's percentage improvement in skill. The formulas are intended to resemble the encirclement of prey mentioned in hunter groups. The new positions of hunters from the left and right wings are as follows, respectively:

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2920 - 2931
https://publishoa.com
ISSN: 1309-3452

$$\text{Hunter}_i = \left(\begin{cases} Random\big((2 \times PreyX - \text{Hunter}), \text{PreyX}\big), \\ \text{Random}(\text{PreyX}, (2 \times \text{PreyX}, (2 \times \text{Preyx} - \text{Hunter})), \\ (2 \times \text{PreyX} - \text{Hunter}) < PreyX \\ (2 \times \text{PreyX} - \text{Hunter}) > Preyx \end{cases} \right)$$

in which PreyX is the predator's new place, Ranger is indeed the hunter's present situation, and Ranger's is the ranger's new location. The following is the new position of the centre Hunter:

$$\text{Hunter}_i = \left(\begin{cases} \{Random, (\text{Hunter}, \text{Preyx}), \\ \text{Random}(\text{PreyX}, \text{Hunter})\text{Hunter}(\text{Preyx}, \text{Hunter}) \\ \text{PreyX}, \text{PreyX}\} \end{cases} \right)$$

The arbitrary numbers produced in the preceding equations rand (a, b), that also lie between 'a' and 'b', where 'a' and 'b' are top and bottom bounds, respectively.

Roaming and mating: The able to roam system enhances local search ability and aids in the discovery of workable answers (task scheduling solution). Evey resident male lion in the pride roams within the land of the pride. During roaming, if the male lion finds a better location than the current location, it needs to update the new spot as the finest frequented location [19]. Furthermore, Ma% of the female lions in the pride P mate with one or more resident male lions, resulting in offspring. Using equations (1) and (2), the mating procedure typically produces two offspring (2).

$$\text{Offspring}_j\ 1 = \beta \times \text{female Lion}j + \sum \frac{(1-\beta)}{\sum_{i=1}^{NR} Si} \times \text{male } Lion_j^i \times S_i \quad (1)$$

$$\text{Offspring}_j\ 2 = (1-\beta) \times \text{female Lion}_j \times \text{female Lion}_j + \sum \frac{(1-\beta)}{\sum_{i=1}^{NR} Si} \times \text{male } Lion_j^i \times S_i \quad (2)$$

In which NR denotes the number of resident males in a pride, is a completely random number with a standard error of 1 and a mean value of 0.1. If Si=1, a male lion is chosen for mating; otherwise, Si=0. The mutation is then performed on the 2 additional offspring, one of whom is randomly identified as female and another as male.

Defense: Males in the pride might very well battle new mature resident males. A most vulnerable males will abandon their pride and become nomads. By combining new mature males and old males, this behaviour can be replicated. The males are then sorted based on their fitness levels. The pride's weakest males are expelled and become nomads, while the remaining males become resident males. This strategy aids our proposed algorithm in retaining strong male lions as important LOA solutions. Nomad males attack prides at random in an attempt to take over a pride by fighting the pride's male lions. If the nomad lion is powerful enough, the weak male lion will be driven out of the pride and become a nomad.

Migration: Influenced by lion switch life and migratory actions in essence, when one lion travels from one pride to another or switches its lifestyle, and the resident female becomes a nomad and vice versa, it increases the diversity of the target pride by its position in the previous pride. The lion, on the other hand, builds a bridge for information exchange through migration and switching lifestyles. The maximum number of females in each pride is determined by S percent of the pride's population. Some females were chosen at random to be nomads by a migration operator. The magnitude of people moved females for each pride is equal to the number of excess females in each pride multiplied by a factor of one of the maximum number of females in a pride. When chosen females migrate from prides to become nomads, they are kept separate into new nomad females and old nomad females based on their athletic ability. The best females from among them are then chosen at random and distributed to prides to fill the migratory females' spaces left. This procedure preserves the diversity of the entire population while also sharing information among prides [20].

III. EXPREMENTAL SETUP

This segment presents the experimental environment for cloud computing. Cloudsim 3.0 is used as a simulation tool. The experiment configuration includes one server with an Intel i5 2.90 GHz processor and 4 GB of RAM. Multiple virtual machines (VMs) can be run on the server. On the server, a VM with a processor of one core, RAM of 512 MB, and secondary storage of one hundred and eighty GB is created and Ubuntu 12.04 is installed as the operating system. The chosen techniques (ACO, PSO, Firefly, and LOA) are developed and constructed on the same experimental platform. The sections that follow make a comparison multiple methods. In the following scenario, the simulation is used: The tasks that must be completed are unrelated. The computational sizes of tasks differ. MI displays the duration of each task

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2920 - 2931
https://publishoa.com
ISSN: 1309-3452

(Millions of Instructions). Initially, 100 tasks and 15 VMs were performed in order to conduct experiments. The parameter settings of the cloud simulator are illustrated in table 1.

Table 1: Parameters Setting of CloudSim.

Type of Entity	Parameters	Values
Task	Length of task	1000-2000(MI)
	Total no of task	100
Virtual Machine	Total no of VMs	15
	MIPS	600-2500
	VM Memory	512-2048
	Bandwidth	80-150
	Number of PEs requirement	1-5
Datacenter	Number of Datacenter	1
	Number of Host	3-7
	VM Scheduler	Time Share and Space shared

Comparison of Cost: As shown in Figure 4, the LOA algorithm does have the lowest cost, the ACO algorithm has the second lowest cost, and the PSO algorithm has the highest cost. Taking into account Firefly's implementation and going to wait costs, as well as the task execution time restriction, the cost of the LOA and ACO algorithms is fairly low. The x-axis represents the number of tasks, while the y-axis represents the cost of task completion per hour. The same results show that the cost of LOA is somewhere between PSO and ACO, but the difference isn't particularly significant. Figure 4 depicts the evaluation metric, which is used to determine the total cost of the task based on the schedule. Each of the four algorithms' maximum task costs are calculated to be 25, 50, 75, and 100. The cost of 25 tasks completed with 10VMs is 123.74, 137.54, and 137.54, in both.143.13 and 156.22 for LOA, ACO, Firefly, and PSO, respectively, with the Firefly coming out ahead. In the second, the costs of 50 tasks are 202.4, 233.15, 249.7, and 260.13 for LOA, ACO, Firefly, and PSO, respectively, with Firefly coming out on top. In the third one, the costs of 75 tasks are 320.15, 364.4, 382.21, and 375 for LOA, ACO, Firefly, and PSO, respectively, with Firefly coming out ahead. In the fourth one, the costs of 100 tasks are 541.8, 574.28, 582.22, and 630.66 for LOA, ACO, Firefly, and PSO, respectively, with Firefly coming out ahead.

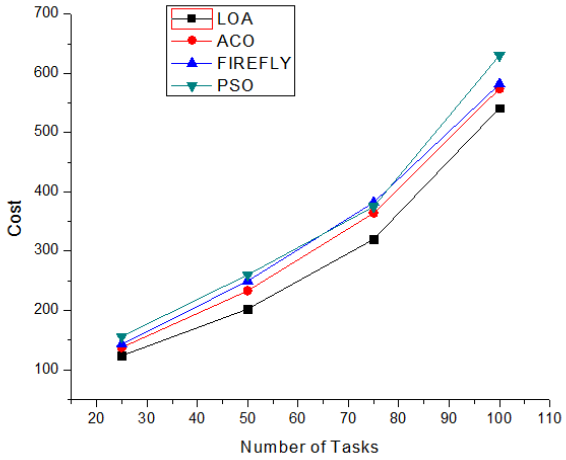

Figure 4: 5 Costs of 10 VMs

Resource Utilization: Figure 5 shows the comparison of average resource utilization between LOA, ACO, PSO and Firefly. It is obvious that PSO provides a very high utilization of resources when compared with PSO and ACO. If RU is the utilization ratio of the VM, then, ***RU (%) = Time Processing Tasks/ Total Time;*** In terms of resource utilization, LOA makes better use of active resources because the migration policy optimises both the number of current hosts and their optimally utilize at the same time.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2920 - 2931
https://publishoa.com
ISSN: 1309-3452

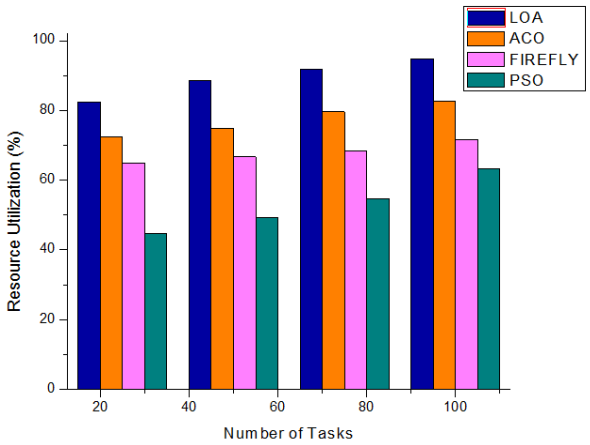

Figure 5: Average Resource Utilization

Execution Time: The experiment is carried out in the first case to evaluate the connection between repetitions and processing time. For all iterations and population sizes of both fireflies and ACO, the number of tasks was fixed at 25. Flows through the search space in PSO, adjusting their position by their own best position or with regard to its friend's best possible position, in search of a best solution. The LOA is organised based on lion actions such as hunting and mating, with the two most basic behaviours being resident and drifting. The resident category is organised into groups known as prides. For all cases, the number of resources or virtual machines (VMs) was set to 5. Starting with 5 iterations, the iterations were increased by factors of 25. The iteration value in fig corresponds to the same iteration value denominated for the LOA, ACO, Firefly, and PSO algorithms in each case.

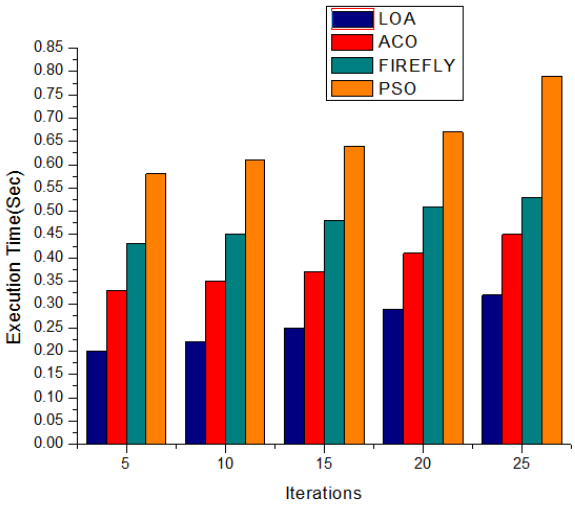

Figure 6: Performance Analysis of Variation between Execution Time and Iterations

Figure 6 depicts an execution time analysis of the LOA model using existing techniques. The results show that the LOA model provides superior performance while requiring the least amount of time to execute. For example, the LOA model achieved a minimum execution time of 0.20 after 5 rounds, whereas the ACO, PSO, and Firefly techniques achieved poor results with execution times of 0.33, 0.43, and 0.58, respectively. Furthermore, after 25 rounds, the LOA model achieved a minimal execution time of 0.32, whereas the ACO, PSO, and Firefly techniques achieved poor results with execution times of 0.45, 0.53, and 0.79, respectively.

Makespan: The total time elapsed from start to finish is referred to as the make span. The term is frequently used in the context of scheduling. There is a large project that is divided into several sub-tasks. Figure 7 depicts a time comparison of ACO, PSO, Firefly, and LOA algorithms. The total time required by the tasks to complete the execution is referred to as the makespan. By varying the task and resource values, Makespan is evaluated and analysed. When compared to ACO, PSO, and Firefly, the LOA algorithm produces the lowest span values. In Figure 7, the LOA method achieves the

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 2920 - 2931
https://publishoa.com
ISSN: 1309-3452

shortest makespan time (1233) while the PSO method achieves the longest (8702) in the randomly generated 50th task scenario.

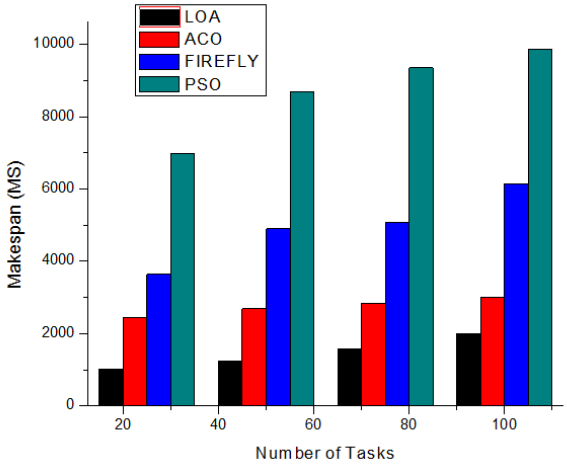

Figure 7: Makespan Time

Completion Time: The time difference between starting and finishing times is referred to as the completion time. When the number of Tasks and VMs is increased, the completion time grows. Figure 8 depicts the iterative process of achieving the best total time and the lowest total cost by combining ACO, PSO, Firefly, and LOA. The LOA algorithm is faster than the ACO, PSO, and Firefly algorithms in terms of completion time. When the number of tasks and VMs is 500, the LOA algorithm completion time is ms and ms faster than the ACO, PSO, and Firefly techniques. Figure 8 shows that the completion time of the four algorithms is the same in the first iteration, but as the number of iterations increases, the convergence rate and accuracy are superior to the scheduling results of LOA, ACO, PSO, and Firefly. This graph is constructed for the 20 tasks and 5 static VMs. If we consider the first entry, the average response time of LOA for the 25th tasks is 59.41, while the average response time of PSO for the same 25th tasks is 112.46. When it comes to efficient task scheduling, the average response time should be less than one hour, according to LOA.

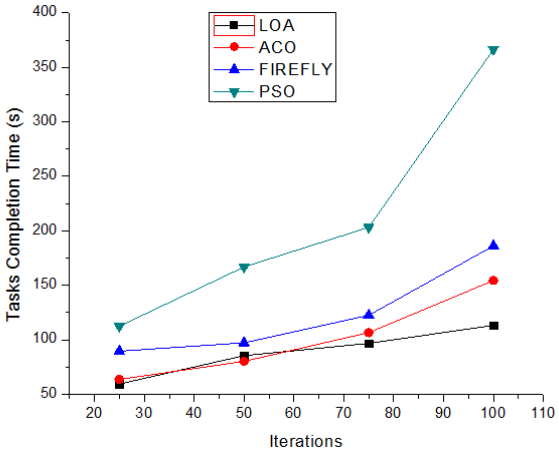

Figure 8: Total time for completing task

IV.CONCLUSION

A variety of scheduling algorithms are available for use in cloud computing environments to improve task and resource scheduling. Scheduling is the process of allocating a specific number of resources to tasks in order to maximize resource utilization while minimizing total processing and waiting time. Task scheduling is the process of mapping user tasks to the appropriate resources for selection and execution. The primary goal of the scheduling technique is to assign tasks to users while minimizing costs and increasing system span. A better task scheduling algorithm is expected to schedule different user's tasks in order to achieve overall system performance with a low cost factor. Experiments show that the

existing ACO, PSO, LOA, and Firefly algorithms satisfy the requirements of users in cloud computing task scheduling effectively.

V.REFERENCES

[1]. Mohammad Hamdaqa and Ladan Tahvildari , "Cloud Computing Uncovered: A Research Landscape". Elsevier Press. pp. 41–85. ISBN 0-12-396535-7.
[2]. Huang Q.Y., Huang T.L.,"An Optimistic Job Scheduling Strat egy based on QoS for Cloud Comput ing", IEEE Int ernat ional Conference on Intelligent Comput ing and Integrated Systems (ICISS), 2010, Guilin, pp. 673-675, 2010
[3]. Cristian Mateos, Elina Pacini & Carlos Garc Garino, (2013), An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments.
[4]. K. Etminani, and M. Naghibzadeh, "A Min-min Max-min Selective Algorithm for Grid Task Scheduling,"The Third IEEE/IFIP International Conference on Internet, Uzbekistan, 2007.
[5]. Rajkumar Buyya, A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments, Cloud Computing and Distributed Systems Laboratory, Department of Computer
[6]. Arfeen M A, Pawlikowski K, Willig A 2011 A Framework for Resource Allocation Strategies in Cloud Computing Environment *Computer Software and Applications Conference Workshops (COMPSACW), IEEE 35th Annual* 261-6
[7]. Kennedy J, Spears W 1998 Matching Algorithms to Problems: an Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator *Proc IEEE International Conference on Evolutionary Computation. Piscataway, NJ: IEEE Service Center* 78-83
[8]. Adil yousif, Abdul hanan abdullah, Sulaiman mohd nor, Adil ali abdelaziz, "Scheduling jobs on grid computing using firefly algorithm" , Journal of Theoretical and Applied Information Technology, ISSN: 1992-8645, 30th November 2011. Vol. 33 No.2
[9]. Yang, X.S., Nature-inspired metaheuristic algorithms. 2010: Luniver Press.
[10]. Fakhrosadat Fanian ," A New Task Scheduling Algorithm using Firefly and Simulated Annealing Algorithms in Cloud Computing" (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 9, No. 2, 2018 19
[11]. Basu, B. and G.K. Mahanti, Fire Fly and Artificial Bees Colony Algorithm for Synthesis of Scanned and Broadside Linear Array Antenna. Progress In Electromagnetics Research, 2011. 32: p. 169-190
[12]. A. Paulin Florence and V. Shanthi," A Load Balancing Model Using Firefly Algorithm In Cloud Computing", Journal of Computer Science 10 (7): 1156-1165, 2014 ISSN: 1549-3636 © 2014 Science Publications
[13]. C. Liu, et al. 2012. A New Path Planning Method Based on Firefly Algorithm. Computational Sciences and Optimization. 23-26 June 2012.
[14]. Aravind Rajagopalan(&), Devesh R. Modale, and Radha Senthilkumar,"Optimal Scheduling of Tasks in Cloud Computing Using Hybrid Firefly-Genetic Algorithm", © Springer Nature Switzerland AG 2020 S. C. Satapathy et al. (Eds.): ICETE 2019, LAIS 4, pp. 678–687, 2020.,pp. 678–687, 2020.
[15]. Stander PE. Cooperative hunting in lions: the role of the individual. *Behav. Ecol. Sociobiol.* 1992;29(6)445–54.
[16]. Nora Ahmed Almezeini , Alaaeldin Hafez"Task Scheduling in Cloud Computing using Lion Optimization Algorithm", *in* International Journal of Advanced Computer Science and Applications · January 2017
[17]. S. Periyanatchi and K. Chitra ,"A Lion Optimization Algorithm for an Efficient Cloud Computing With High Security", Journal of Scientific Research, Volume 64, Issue 1, 2020
[18]. Jagmeet Kaur, Shakeel Ahmed, Yogesh Kumar, A. Alaboudi, N. Z. Jhanjhi and Muhammad Fazal Ijaz," Packet Optimization of Software Defined Network Using Lion Optimization, *Computers,Materials & Continua*, Received: 31 January 2021; Accepted: 13 April 2021
[19]. A.Tamilarasi, A.Abarna, K.Chitra, K.Nagendhiran,R.Aarthi, " Effective Data Clustering Using K Means Along with Lion Optimization Algorithm" ISSN: 2005-4238 IJAST, Copyright, 2020 SERSC
[20]. A. Kaveh, S. Mahjoubi,"Lion pride optimization algorithm: A meta-heuristic method for global optimization problems", Sharif University of Technology Scientia Iranica Transactions B: Mechanical Engineering