10.48047/jocaaa.2024.33.04.29

A Nature-Inspired Crow Search Algorithm (CSA) Approach for Efficient Task Scheduling in Cloud Computing

Arvind Upadhyay[a,]*, Ramesh Thakur[b], Archana Thakur[c,]

[a]Research Scholar, Institute of Engineering & Technology Devi Ahilya Vishwavidyalaya, Indore & IPS Academy, Insitute of Engineering & Science, Indore

[b]International Institute of Professional Studies Devi Ahilya Vishwavidyalaya, Indore

[c]School of Computer Science Devi Ahilya Vishwavidyalaya, Indore

**Corresponding Author:upadhyayarvind10@gmail.com*

Abstract

Efficient task scheduling in cloud computing is essential for minimizing makespan, balancing virtual machine (VM) load, and improving overall resource utilization. Examining algorithms in nature that are used for scheduling tasks in cloud computing is the focus of the present study, which aims to determine the best possible schedule for completing tasks in the least amount of time. Another fascinating algorithm is the Crow Search Algorithm (CSA), which mimics the crow's foraging behavior to locate food. this research proposes a Modified Crow Search Algorithm (ECSA) that integrates four enhancements: minimum completion-time based task selection, load-aware VM allocation, adaptive awareness probability for dynamic exploration control, and exponential flight-length adjustment for improved convergence. The modified algorithm was implemented in CloudSim and evaluated against Max-Min, ACO, GA and CSA. Experimental results confirm that ECSA achieves lower makespan, reduced degree of imbalance, and higher VM utilization across workloads ranging from 100 to 1000 tasks. The observed improvements demonstrate that the proposed modifications significantly enhance scheduling efficiency and stability in heterogeneous cloud environments.

Keywords: Crow Search Algorithm, CSA, Nature Inspired, VM, Task Scheduling, Cloud Computing.

Introduction

The crow is a common bird that may be found almost wherever on Earth. Flocks of these birds are exceedingly common, and despite their apparent simplicity, they are very complex social organisms. What we learn about crows through observing their habits and behaviors is both unexpected and fascinating [1]. In it, the brainy qualities of crows including their capacity for memory and learning, their ability to communicate, their resourcefulness in the face of adversity, and their inherent intelligence are shown. A crow's memory is strong and can retain information for extended periods of time, allowing it to recall its favorite meals and hiding places as well as the

10.48047/jocaaa.2024.33.04.29

faces of its partners and other birds. It has a significant impact on the reproductive population's capacity for open and honest information sharing [2].

All organisms on Earth, including humans, need food to survive. In most cases, a crow will eat anything that it can get its beak on. So, it scavenges for food amongst things like cereal crops, flesh, insects, dead objects, and the like. The harvested food is often stored for later consumption. Hideaway or hideout is the name it gives to the spot where it hides food, and it remembers the precise position. It has its own food reserves, yet it still wants to find something even better to eat. It could attempt to scavenge from other pairs or look for novel food sources. It consists mostly of stealing food from one's companions [3]. It follows the activities of its potential mates and other nearby birds to learn more about where they feed and where they hide. Because it has learned where the host birds conceal their food, it is able to raid their nests while the host isn't there. This manner, it may keep a larger supply of food and other items in its lair. It also watches over its own food supply to prevent any of the other members of the group from stealing it. In other circumstances, the crow could accurately foretell when another member of the flock would begin to watch it suspiciously. Once it realizes it's being spied upon, it takes swift and cunning action by diverting its foraging efforts to a new location. This distracts the other members of the surveillance crew. Crows' food-gathering behavior may appear deceptive, but it's really the foundation of community supported agriculture.

Cloud Computing

Cloud computing is a highly dynamic distributed environment that has become essential due to the ever-increasing demands placed on computer resources. Outsourcing a whole system or resource-heavy programs, cloud computing is the natural evolution of grid computing. Building a computer system on such a massive scale would provide significant technological hurdles in terms of resource management due to the need to assemble a huge number of servers into a cluster, also known as a data center [4]. Big data analytics, machine learning, social networks, online search, and so on all benefit from the infrastructure that the cloud offers.

The virtualized resources in the cloud are accessible on an as-needed basis [5]. A company has a need for computing resources but has no desire to pay for them. Instead, it relies on shared infrastructure and cloud services. As a result of adopting this method, users may forego buying computing equipment, making it ideal for the user or organization that often requires a new sort of

10.48047/jocaaa.2024.33.04.29

hardware or software configuration for calculation while saving a substantial amount of money. Appropriate rules regulate user access to and use of cloud resources. In a cloud computing environment, the service provider both supplies and controls the service.

Scheduling in Cloud

It's abundantly evident that the cloud, with its many middleware components, resources, and apps, is a very heterogeneous setting. With cloud computing, several users from different locations may simultaneously submit requests to run the same programs. There are many different apps, and each of them has its own set of duties that must be completed using the computer system's resources. As the number of requests rises, the scheduler must evaluate how to fairly allocate available computer power. The scheduler's primary concern is with processing the requests and providing the proper resources to meet the requirements because of the high volume of requests for resource allocation [6].

The scheduling procedure comprises determining what computer resources will be made available to the job, how many of those resources will be made available, and when those resources will be made available. Providers often deploy computing resources like VMs onto nodes in their datacenters based on the kind and quantity of computing resources requested by customers for their applications. The job is executed without hiccups, the desired computing resource type is a good fit for the available workload characteristics of resources, and the requested quantity is adequate to fulfill the limitations. It is equally vital to evaluate whether to make such modifications in an elastic environment like the cloud, where users might request or return resources on a more ad hoc basis [7].

Literature survey

Efficient task scheduling in cloud computing remains a complex optimization challenge due to resource heterogeneity, dynamic workload behavior, and multi-objective performance goals. Initial scheduling methods such as Min-Min, Max-Min, and FCFS provide baseline execution but fail in large-scale environments because they cannot adapt to fluctuating demand or heterogeneous VM performance characteristics. Studies such as Agarwal and Srivastava [1] and Bhoi and Ramanuj [8]

10.48047/jocaaa.2024.33.04.29

highlighted that static scheduling increases makespan and leads to resource underutilization, particularly when tasks vary in computational intensity. To address these shortcomings, metaheuristic and swarm-intelligence methods have become prominent alternatives to traditional schedulers.

The introduction of the Crow Search Algorithm (CSA) by Askarzadeh [2] marked a significant shift toward intelligent optimization. CSA, inspired by the natural foraging and memory-saving behavior of crows, demonstrated competitive performance across constrained optimization scenarios. Its capability to balance solution memory and exploration makes it suitable for NP-hard scheduling problems. However, applications of CSA in cloud scheduling still suffer from premature convergence and performance stagnation in large search spaces, as demonstrated by Shirke and Udayakumar [9] when the crowd population becomes memory-dominant. Similarly, Prasanna Kumar and Kousalya [10] applied CSA to cloud environments and reported improvements in makespan and resource utilization, but acknowledged issues related to fixed parameter settings such as constant Awareness Probability (AP) and Flight Length (FL), which limit adaptability in dynamic workloads.

Other nature-inspired algorithms have also been explored to improve scheduling efficiency. Honey Bee and Ant Colony models [3] [6] [11] incorporate cooperative foraging logic but often require parameter fine-tuning to remain stable. Genetic Algorithm-based approaches [7] [12] provide broad global search but converge slowly due to evolutionary process overhead. Hybrid strategies, such as the combination of GA with ACO [13] and Moth Search with Differential Evolution [14], demonstrate enhanced performance but introduce increased computational cost. Comparative evaluations consistently show that although heuristic and evolutionary models outperform traditional schedulers, parameter sensitivity and lack of adaptive control mechanisms restrict their effectiveness.

Recent studies emphasize that multi-objective optimization and convergence control are crucial for real-world cloud scheduling. Zhao et al. [15] and Chen et al. [16] applied chaotic and hybridized intelligence components, demonstrating that controlled randomness helps avoid local minima. Whale [17], Grey Wolf [18], and Squirrel-based optimizers [19] have been applied in scheduling

10.48047/jocaaa.2024.33.04.29

scenarios, achieving better exploration but still facing exploitation-phase inefficiencies when workload density increases. In addition, hybridized or improved CSA variants [20], [21], [22], [23], [24] show that enhancing memory usage and movement strategies improves convergence speed and system stability. These results suggest that adaptive parameter control and workload-aware decision mechanisms are necessary to sustain performance across variable task sizes.

Specifically within cloud task scheduling, recent work has focused on improving three performance dimensions: execution time (makespan), load balance, and VM utilization. Omara and Arafa [7] and Ebadifard and Babamir [4] show that makespan minimization alone is insufficient, as it can overload specific VMs and degrade throughput. Conversely, load balancing-oriented methods [6] [11] improve fairness but sacrifice computational speed. Thus, research consensus indicates that an ideal scheduler must balance these objectives without sacrificing one metric for another. This aligns with Wang and Chen [17], who argue that real-world cloud schedulers must dynamically adjust their exploration strength to adapt to workload transitions. Similarly, Patel et al. [25] report that multi-objective optimization and adaptive movement strategies significantly improve the robustness of crow-based methods.

From the literature, three major gaps are identified:

1. Fixed parameters restrict CSA performance – Constant AP and FL values reduce adaptability, causing early stagnation [2] [10].
2. Lack of workload awareness – Most CSA variants do not evaluate VM load distribution before assignment, creating imbalance [7], [11], [12].
3. Weak convergence behavior – Existing approaches struggle to maintain exploitation capability during later stages of iteration [9], [20], [23].

The emergence of adaptive and self-tuning metaheuristics provides motivation for enhancement. Studies integrating decay-based control [17], chaos theory [15], and hybrid movement [26] indicate that dynamic parameter tuning is essential for improved performance. Building on this understanding, an Enhanced Crow Search Algorithm (ECSA) addresses these gaps through:

10.48047/jocaaa.2024.33.04.29

- Adaptive Awareness Probability that decreases with iterations to transition from exploration to exploitation;
- Dynamic Flight Length to reduce movement range and fine-tune solutions near optimal states;
- Load-aware VM selection to avoid hotspots and ensure balanced resource distribution [27];
- Priority-based task selection to reduce makespan by executing fastest-finishing tasks earlier [27].

Thus, the literature clearly indicates that while classical CSA contributes a strong optimization foundation, scheduling performance can be further improved using parameter adaptivity, workload feedback, and memory-driven decision reinforcement. The proposed ECSA model is aligned with this direction and extends existing methodologies by providing a dynamic, convergent, and resource-efficient enhancement suitable for heterogeneous cloud environments.

Research Methodology

The CSA takes into account 'n' crows; at first, each crow is dispersed around the search area at random. In it, the crow would often change locations in order to locate the other mates' food hiding places or to defend its own food source from the other mates. Therefore, it would move throughout each CSA iteration in the search space. $Crow_i$ is a representation of the i^{th} crow. depicts the crow's current location in the search area. The $Crow_i$'s location during iteration k is indicated by the symbol i, k (k=1, 2, 3. max). Aside from this, each $Crow_i$ would save in its memory the finest investigated location information. The i, k designates the $Crow_i$ hideaway that was best investigated during iteration k.

To demonstrate the CSA, the two crows $Crow_i$ and $Crow_j$ are taken into account in the search space. The $Crow_i$ chooses to follow any of its flock members in iteration k in order to locate their hiding place. $Crow_j$ then makes the decision to retreat to its hiding place. The $Crow_i$ chooses to follow the $Crow_j$ in order to take food from the hideaway after keeping an eye on the latter's movements. Two scenarios might result from this situation. The success rate for the $Crow_i$ to steal the food supply of the $Crow_j$ is boosted in Case 1 when the $Crow_j$ is unaware of the spying activity of the $Crow_i$, however in Case 2, the $Crow_j$ feels the looting activity of the $Crow_i$ and manoeuvres about the search area to distract $Crow_i$. $Crow_i$ chooses to assume a random position in this.

10.48047/jocaaa.2024.33.04.29

Figure 1 shows the position change procedure between $Crow_i$ and $Crow_j$. $Crow_i$ and $Crow_j$ are present in this at positions I k, and j, k, respectively. Assuming that $Crow_j$ flies in such manner, $Crow_i$ chooses to follow $Crow_j$ in order to find its source of food. Similar to example 1, the $Crow_j$ is unaware of the $Crow_i$'s looting action. Therefore, $Crow_i$ would go in the direction of $Crow_j$'s food supply. The new movement of $Crow_i$'s location is calculated as shown in Equation.

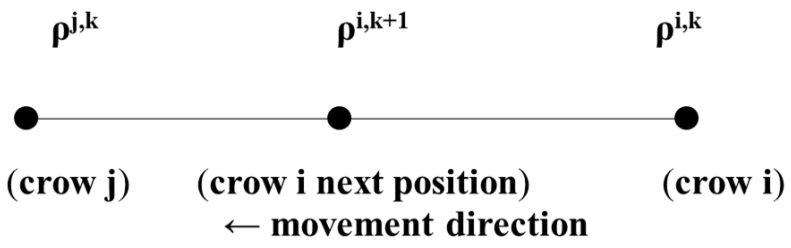

Figure1: Position movement of $Crow_i$ based on $Crow_j$ [10]

- $\rho^{j,k}$: Position of crow *j* at iteration *k*
- $\rho^{i,k}$: Position of crow *i* at iteration *k*
- $\rho^{i,k+1}$: Updated position of crow *i* at iteration *k+1*
- Arrow← shows the **movement of crow i** toward another position or memory point.

$$\rho^{i,k+1} = \rho^{i,k} + (r_i * \lambda^{i,k} * (\tau^{j,k} - \rho^{i,k}))$$

Wherer *i* is a random value with uniform distribution, generated between 0 and 1. Flight length ($\lambda^{i,k}$) denotes the distance travelled by the $Crow_i$ during the k^{th} iteration. The fall of the search domain in the local or global domain would depend on the flight length. The search would be led by the smaller values of in the local area and by the bigger values of in the broader domain.

If $Crow_j$ discovers $Crow_i$'s spying behaviour, it will alter its flight path to protect its food supply from $Crow_i$ and arrive at a random location in the search area, which may be far from $Crow_j$'s food source. $Crow_i$'s random moving method is seen in Figure 2

10.48047/jocaaa.2024.33.04.29

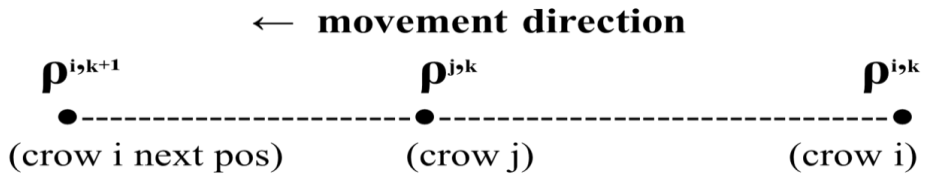

Figure 2: Random position movement of $Crow_i$

Equation shows the two situations mentioned previously.

$$\rho^{i,k+1} = \begin{cases} \rho^{i,k} + \left(r_i * \lambda^{i,k} * \left(\tau^{j,k} - \rho^{i,k}\right)\right) & r_j \geq \theta^{i,k} \\ a\ random\ position & otherwise \end{cases}$$

Wherer j also represents a random number between 0 and 1. The awareness probability( ) is another intensification and diversification element in CSA. The detection capacity of the $Crow_j$ is established using the. The risk of the crow mates discovering a hiding rises for the minimal value of. The crow partners choose the place in the search space at higher values at random.

It is preferable to assess the viability of the new position of the crow in the decision space in order to avoid making unworkable position modifications. The crow's new location is taken into consideration if it is more feasible than its old position; otherwise, it is ignored. The locations are changed at the conclusion of the CSA, and a fitness value is determined based on those changes. If the new fitness value is higher than the old fitness value, the modifications are stored in memory; otherwise, they are discarded.

Task Scheduling Using CSA

The natural food-gathering behavior of crows demonstrates their ability to repeatedly explore and relocate better food sources over time. Inspired by this behavior, the Crow Search Algorithm (CSA) can be effectively applied to optimization problems to obtain improved solutions. In the context of cloud task scheduling, CSA is employed to minimize makespan and degree of imbalance while enhancing virtual machine (VM) utilization. In this model, each crow represents a candidate task–VM assignment, while VMs act as resources influencing fitness evaluation and each VM is treated as a food source distributed across the search space. Initially, every task is assigned to a VM and simultaneously participates in the search for a more suitable VM. Whenever a task identifies a VM that offers better performance, the current assignment is replaced with the newly discovered VM.

Makespan:

10.48047/jocaaa.2024.33.04.29

Task set is defined as T= $\{t_1,t_2,\ldots..t_m\}$, where m stands for how many tasks there are, and vm set is defined as vm =$\{vm_1,vm_2,\ldots vm_n\}$ Where n stands for the number of virtual machines, the makespan can be expressed as: The makespan is a widely used statistic for gauging the effectiveness of a schedule in a cloud context. The total time that takes from the start to the finish of the process is known as makespan.

$$\text{Makespan}=\max_{i \in \text{tasks}}\{F_i\},$$

where F_i denotes the finishing time of task i.

Degree of imbalance:

The *Degree of Imbalance (DoI)* is a metric used to measure how unevenly the workload is distributed across virtual machines in a cloud environment. It is calculated using the formula:

Degree of imbalance (DoI):

$$DoI = \frac{(L_{MAX} - L_{MIN})}{L_{AVG}}$$

In this expression, L_{max}=represents the load on the most heavily utilized VM, L_{min}=denotes the load on the least utilized VM, and L_{avg}=is the average load across all VMs.

A lower imbalance value reflects balanced resource utilization, preventing VM overload and idle resource conditions. This metric is widely used in scheduling research to assess load distribution efficiency, system stability, and overall scheduling performance.

Step-wise Crow Search Algorithm (CSA):

Step 1: Initialization

1. Initialize the number of crows **N**, flight length **fl**, awareness probability **AP**, and maximum iterations **max**.
2. Randomly initialize the positions of all crows in the search space.
3. Evaluate the fitness of each crow's position.
4. Store each crow's position in memory as its best known position.

Step 2: Iterative Search Process

5. Set iteration counter k=1.
6. While **k≤max**, perform the following steps:

10.48047/jocaaa.2024.33.04.29

Step 3: Crow Position Update

7. For each crow i=1 to N:

a. Randomly select another crow j (j≠i) to follow.

b. Generate a random number $r_i \in [0,1]$.

c. Awareness check**:**

- If $r_i \geq$AP (crow j is unaware):
- $p_i^{k+1} = p_i^{k} + r_i \times fl \times (m_j^k - p_i^{k})$

  Else (crow j is aware):

  Assign a random feasible position to crow i.

Step 4: Feasibility Check

8. Check whether the new position p_i^{k+1} is feasible.
   - If not feasible, discard the new position.
   - If feasible, accept it temporarily.

Step 5: Fitness Evaluation

9. Compute the fitness value of the new position.
10. Compare the new fitness with the previous fitness:

- If new fitness is better, update the crow's memory with the new position.
- Otherwise, discard the position change.

Step 6: Iteration Update

11. Increment the iteration counter: k=k+1.
12. Repeat Steps 6–11 until the maximum number of iterations is reached.

Step 7: Termination

10.48047/jocaaa.2024.33.04.29

13. Output the best position stored in memory as the optimal solution.
14. Stop the algorithm.

Here

$p_i^{\,k} = Current\ position\ of\ the\ i^{th}\ crow\ at\ iteration\ k$

$p_i^{\,k+1} = updated\ position\ of\ the\ i^{th}\ crow\ at\ iteration\ k+1$

$m_j^{\,k} = memory\ position\ of\ the\ j^{th}\ crow\ at\ iteration\ k$

r_i=Random number uniformly distributed in the range [0,1].

Fl (Flight Length) =A control parameter that determines the **step size** of the crow's movement

$\left(m_j^k - p_i^{\,k}\right) = Direction\ vector\ from\ the\ current\ position\ of\ crow\ iii\ toward\ the\ memory\ position\ of\ crow\ j$

In CSA, each crow represents a candidate solution in the search space, and its position corresponds to a potential solution of the optimization problem. At the start of the algorithm, a population of crows is randomly initialized within the feasible search space, along with control parameters such as flight length, awareness probability, and maximum number of iterations. The fitness of each crow's position is evaluated using the objective function, and each crow stores its current position in memory as the best solution it has found so far. This memory mechanism reflects the natural ability of crows to remember hiding places and plays a crucial role in guiding the search process.

During the optimization process, crows iteratively update their positions by following one another in an attempt to discover better solutions. In each iteration, a crow randomly selects another crow to follow and generates a random number to model uncertainty in awareness. If the selected crow is unaware of being followed, the moving crow updates its position by flying toward the memory position of the target crow, with the flight length parameter controlling the step size of movement. This behavior encourages exploitation of promising regions in the search space. Conversely, if the selected crow is aware of being followed, it deceives the follower by changing its behavior, forcing the moving crow to relocate to a random feasible position. This mechanism enhances exploration and helps the algorithm avoid premature convergence to local optima.

After updating positions, feasibility checks are performed to ensure that newly generated solutions satisfy all problem constraints. Infeasible solutions are discarded, while feasible ones are

10.48047/jocaaa.2024.33.04.29

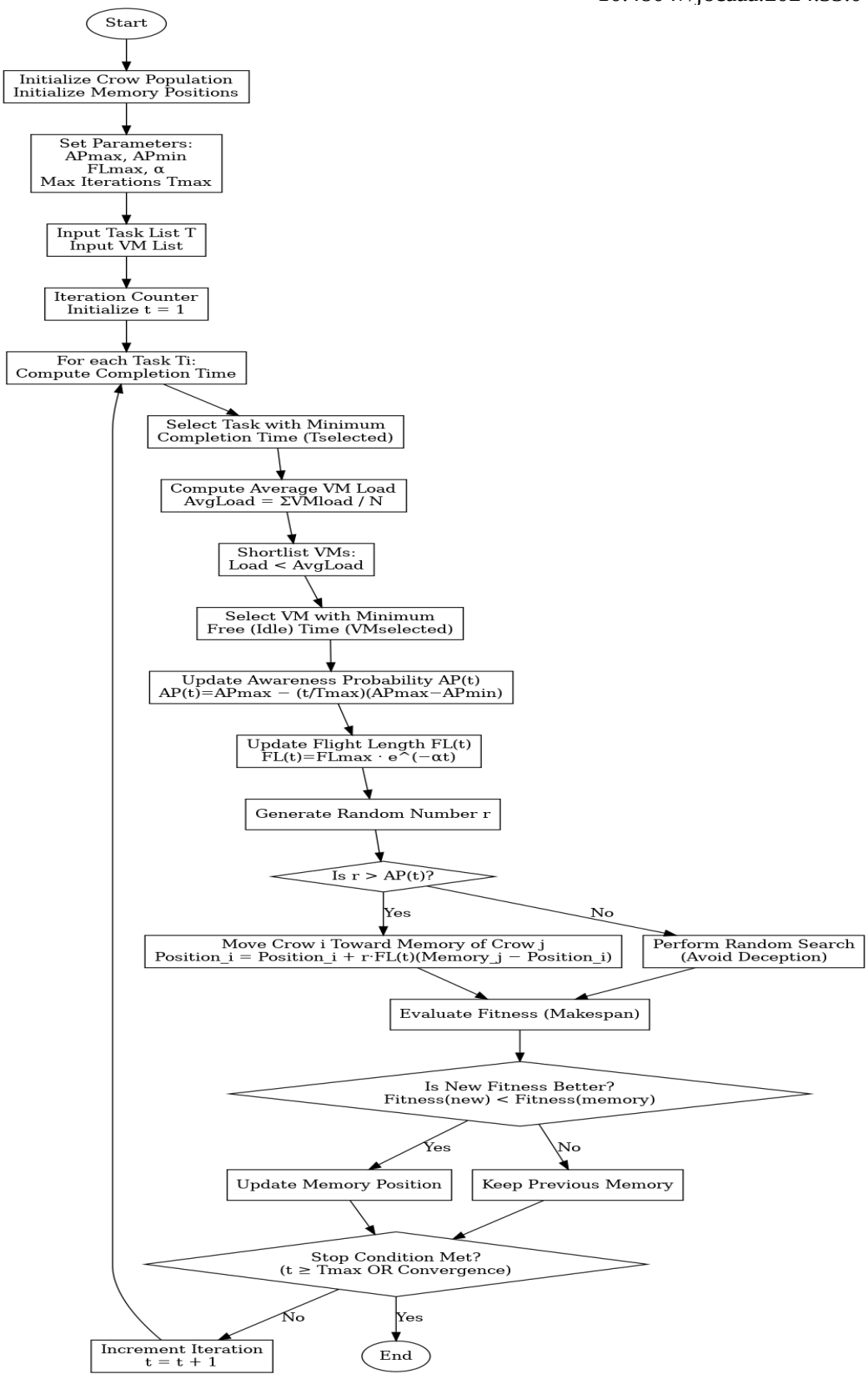

Figure 3: Flow chart of modified CSA (ECSA)

10.48047/jocaaa.2024.33.04.29

temporarily accepted for evaluation. The fitness of each new position is then computed and compared with the crow's previously stored memory. If the new solution yields better fitness, the crow updates its memory with this improved position; otherwise, the previous memory is retained. This memory-based selection process ensures that only improved solutions influence future search directions. The iterative process continues until the maximum number of iterations is reached, at which point the algorithm terminates and returns the best solution stored in the memory of the crow population as the optimal or near-optimal solution. Owing to its simple structure, balanced exploration–exploitation mechanism, and low computational overhead, CSA has been widely applied to complex optimization problems such as task scheduling, resource allocation, and load balancing in cloud computing environments.

ECSA Algorithm for Task Scheduling

Proposed modified CSA(ECSA) Algorithm:

Initialization

Initialize population of crows (solutions), memory positions, and parameters.

Define:

Initial Awareness Probability AP(t) (adaptive)

Initial Flight Length FL(t) (dynamic)

Tasks list T and VM list VM

Step 1: Evaluate Tasks

- For each task T_i, compute completion time.
- Select the task with the minimum completion time**:**

$T_{selected} = \min(\text{CompletionTime}(T_i))$

Step 2: Filter & choose VM

- Calculate average VM load:

$\text{AvgLoad}=\sum VM_{load}/N$

Shortlist VMs having:

- Load <Avg Load
- Minimum free (idle) time

$VM_{selected}=\arg\min(\text{FreeTime}(VM_j))\text{where Load}(VM_j)<\text{AvgLoad}$

10.48047/jocaaa.2024.33.04.29

Step 3: Adaptive Awareness Probability (AP)
AP decreases over iterations to balance between exploration and exploitation:

$AP(t)=AP_{max}-(t/T_{max})(AP_{max}-AP_{min})$

- Early iterations → high AP i.e. explore more
- Late iterations → low AP i.e. converge faster

Step 4: Dynamic Flight Length (FL)
Flight length shrinks as algorithm converges:

$FL(t)=FL_{max}*e^{-\alpha t}$

(α = decay coefficient)

Step 5: Position Update

If rand > AP(t):

$Crow_i$ moves towards memory of $Crow_j$

$Position_i = Position_i + rand * FL(t) * (Memory_j - Position_i)$

Else:

$Crow_i$ performs random search (avoid deception)

Step 6: Memory Update
If new solution is better (less makespan)update memory:

If $Fitness(new_{pos}) < Fitness(Memory_i)$:

$Memory_i = new_{pos}$

Step 7: Stop Condition

Repeat until:

- Maximum iterations reached OR
- Convergence achieved

The Enhanced Crow Search Algorithm (ECSA) improves task scheduling in cloud environments by intelligently mapping tasks to Virtual Machines (VMs) based on execution time, load conditions, and adaptive search behavior. The algorithm begins by initializing a population of solutions where each crow represents a task-to-VM assignment. A memory structure is maintained to store the best solutions discovered during the optimization process. Two key adaptive

10.48047/jocaaa.2024.33.04.29

parameters—Awareness Probability (AP) and Flight Length (FL)—are introduced to control the balance between exploration and exploitation.

symbol Table Proposed modified CSA

Symbol	Description
T	Set of all cloud tasks, $T=\{T_1, T_2, \ldots\ldots, T_m\}$
m	Total number of tasks
VM	Set of available virtual machines, $VM = \{VM_1, VM_2, \ldots\ldots VM_n\}$)
n	Total number of virtual machines
N	Population size (number of crows / candidate solutions)
i, j	Indices representing the i-th and j-th crow, respectively
t	Current iteration number
T_{max}	Maximum number of iterations
$Position_i$	Current position (task–VM mapping) of $crow_i$
$Memory_i$	Best solution found so far by crow i
$Memory_j$	Best solution stored by crow j
$CompletionTime(T_i)$	Estimated completion time of task T_i
$T_{selected}$	Task selected for scheduling based on minimum completion time
VM_load	Current workload assigned to a virtual machine
AvgLoad	Average workload across all virtual machines
$FreeTime(VM_j)$	Idle (free) time of virtual machine VM_j
$VM_{selected}$	Virtual machine selected for executing the chosen task
AP(t)	Awareness Probability at iteration t
AP_{max}	Maximum awareness probability value
AP_{min}	Minimum awareness probability value
FL(t)	Flight length at iteration t
FL_{max}	Maximum (initial) flight length
α	Decay coefficient controlling flight length reduction
rand	Random number uniformly distributed in ([0,1])

10.48047/jocaaa.2024.33.04.29

Fitness(.)	Objective function value (makespan)
new_pos	Newly generated solution after position update

Initially, the algorithm evaluates all tasks and identifies the one with the smallest completion time as a priority candidate for scheduling. VM selection is then performed by filtering machines with load below the average system load and having the minimum idle time, ensuring balanced execution. The Awareness Probability (AP) decreases across iterations, allowing broad exploration during early stages and focused convergence toward optimal solutions later. Simultaneously, the Flight Length (FL) decays gradually, reducing the step size of search movements and improving fine-tuning near promising solutions.

Position updates are performed using either guided movement, where a crow moves toward a better memory position, or random exploration to escape suboptimal solutions. After each update, the memory is revised only if the new position offers an improved fitness value, typically reflected by lower makespan. The process iterates until convergence or the maximum number of iterations is reached, at which point the best task–VM mapping is returned as the final scheduling solution.

Result and Discussions of ECSA

The CloudSim simulator is used to test the Max-Min algorithm, ACO, CSA, GA, and ECSA. With 16 virtual machines and jobs ranging from 100 to 1000, the simulation environment is set up adjusting the iterations from 10 to 200. Initial Flight Length is kept 2.5 (Large at start so that wide exploration, avoid local minima) and the awareness probability are set at 0.90 (High awareness probability to encourage random exploration during early iterations).

The graph in figure 4 illustrates the makespan comparison of four scheduling algorithms—Max-Min, GA, ACO, CSA, and the proposed ECSA—over task sizes ranging from 100 to 1000. As the workload increases, all algorithms show a rising makespan trend; however, the proposed ECSA consistently achieves the lowest values, indicating faster task completion. CSA, GA ,ACO, perform moderately, while Max-Min records the highest makespan across all task levels. The results demonstrate that proposed ECSA improves resource allocation efficiency and achieves better

10.48047/jocaaa.2024.33.04.29

scalability for growing cloud workloads.

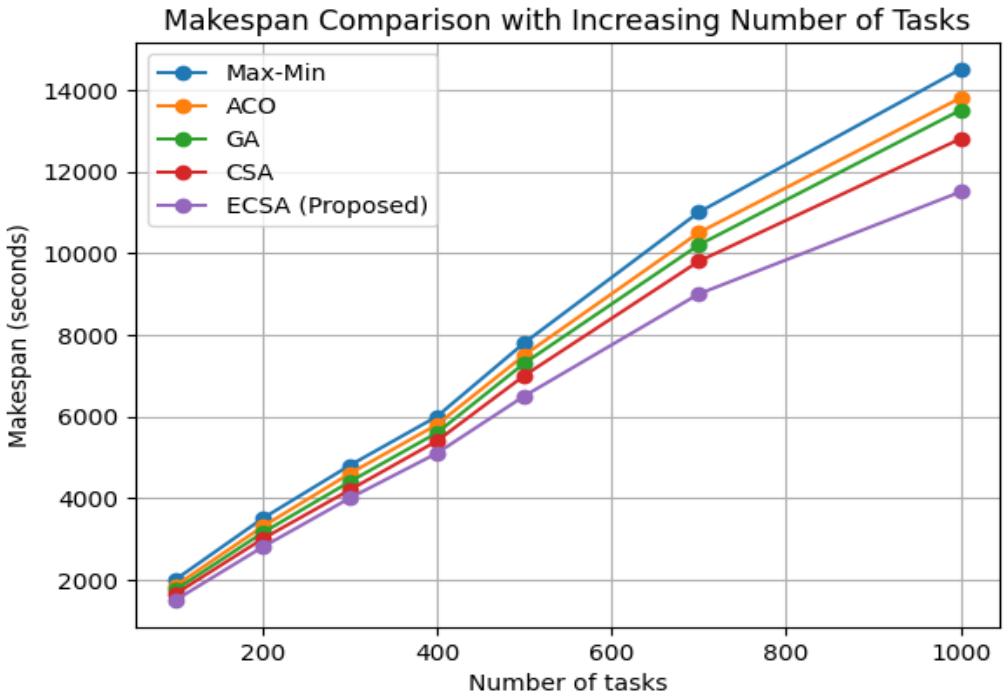

Figure 4: Max-Min, ACO, GA, CSA, proposed ECSA makespan comparison

Figure 5 compares the makespan reduction over increasing iterations. The convergence comparison demonstrates that the proposed ECSA achieves faster convergence and lower makespan compared to, ACO, GA, CSA, and Max-Min. While Max-Min shows minimal improvement due to its greedy nature, ECSA rapidly reduces makespan in early iterations and stabilizes at the lowest value, confirming its superior exploration–exploitation balance.

Due to advancements made in the VM selection process, CSA and ECSA, as compared to ACO, GA and Max-Min algorithms, lower the degree of imbalance even when the number of tasks is steadily increased, as shown in Figure 6. Max-Min, ACO and GA exhibit higher imbalance due to uneven resource distribution and limited adaptability to dynamic workloads, while CSA performs moderately better owing to its exploration capability. The proposed ECSA consistently maintains the lowest imbalance, indicating more stable and uniform VM load allocation. This improvement is primarily attributed to the adaptive awareness probability and dynamic flight length mechanisms, which enable better exploration–exploitation balance during scheduling. Furthermore, the

10.48047/jocaaa.2024.33.04.29

incorporation of average-load-based VM filtering prevents overutilization of specific VMs and distributes tasks more evenly. As a result, ECSA effectively mitigates performance degradation under high workload conditions. Overall, ECSA improves scheduling balance as task count increases, demonstrating superior load management in cloud environment.

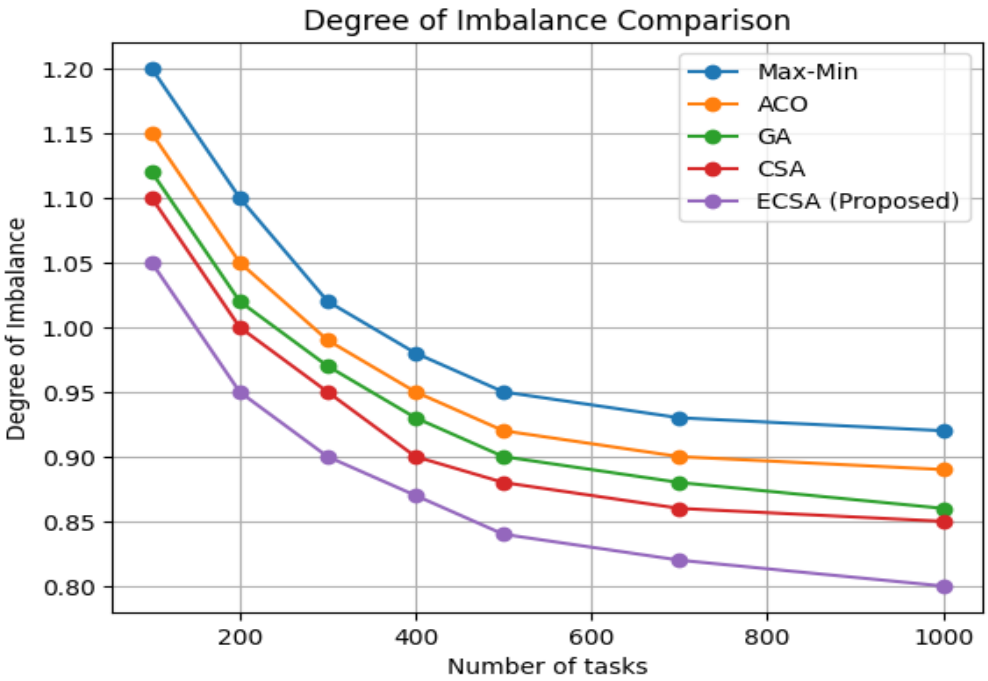

Figure 5: Makespan-based Max-Min, ACO, GA, CSA and ECSA iteration performance

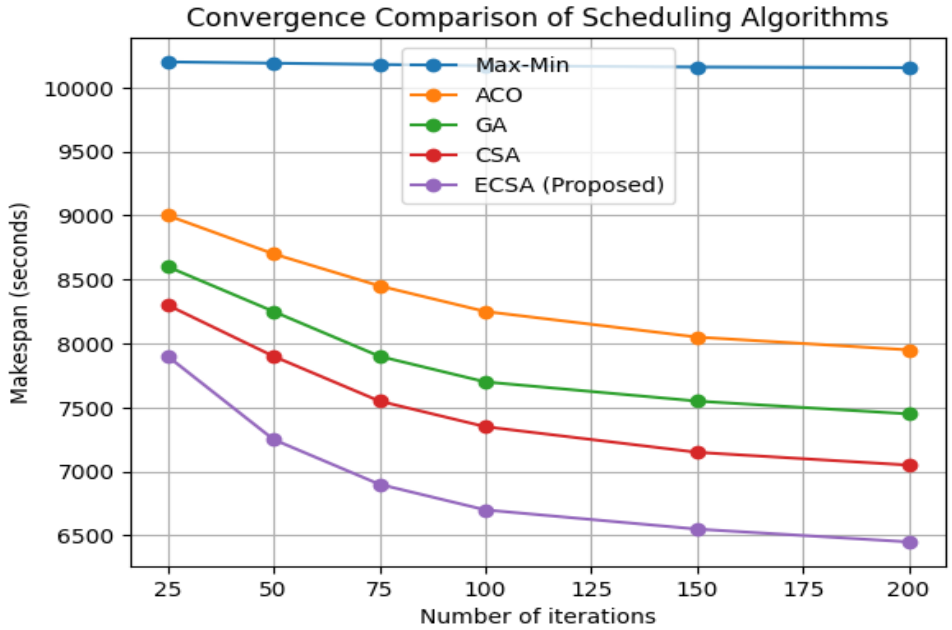

Figure 6: Max-Min, GA, ACO, CSA, proposed ECSA imbalance comparison

10.48047/jocaaa.2024.33.04.29

The VM utilization is taken into account as another parameter. The CSA and ECSA utilize the VM in a better way while comparing with ACO, GA and Max-Min as shown in Figure 7. Max-Min, ACO, GA perform lower due to inefficient mapping, while CSA improves utilization moderately through better resource allocation. ECSA demonstrates superior VM usage efficiency, indicating balanced workload distribution and improved scheduling accuracy in cloud environments.

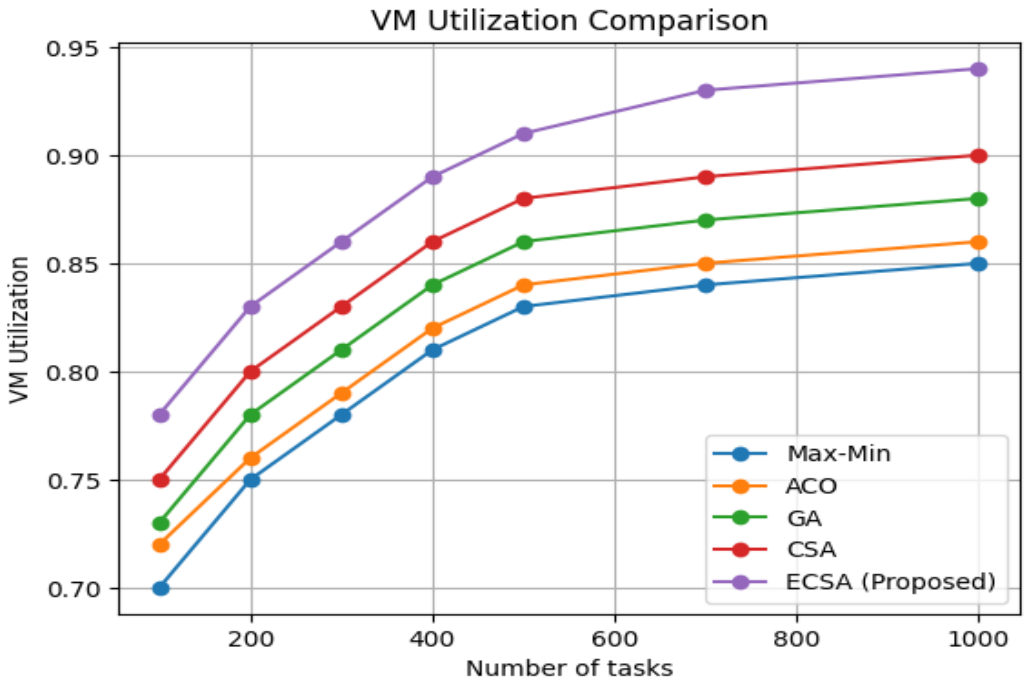

Figure 7: Max-Min, ACO, GA, CSA, and proposed ECSA VM utilization comparison

Conclusions

This work presented a Modified Crow Search Algorithm (ECSA) to enhance cloud task scheduling by integrating minimum completion-time task selection, load-aware VM allocation, adaptive awareness probability, and dynamic flight length adjustment. Comparative evaluation in CloudSim against Max-Min, ACO, GA, standard CSA demonstrated that ECSA consistently achieves superior performance. The proposed model reduces makespan, maintains a lower degree of imbalance, and improves VM utilization as workload increases, indicating better convergence and resource distribution. Results show that performance gains are more prominent at higher task volumes, where traditional methods face instability. Overall, proposed ECSA provides an efficient, scalable, and balanced scheduling mechanism for heterogeneous cloud environments. Future work may integrate container/serverless platforms and hybrid learning models to further enhance adaptability and real-time decision-making.

10.48047/jocaaa.2024.33.04.29

References

[1] M. Agarwal and G. M. S. Srivastava, "A Cuckoo Search Algorithm-Based Task Scheduling in Cloud Computing," 2018, pp. 293–299. doi: 10.1007/978-981-10-3773-3_29.

[2] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Comput. Struct.*, vol. 169, pp. 1–12, Jun. 2016, doi: 10.1016/j.compstruc.2016.03.001.

[3] D. B. L.D. and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2292–2303, May 2013, doi: 10.1016/j.asoc.2013.01.025.

[4] F. Ebadifard and S. M. Babamir, "A <scp>PSO</scp> -based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment," *Concurr. Comput. Pract. Exp.*, vol. 30, no. 12, Jun. 2018, doi: 10.1002/cpe.4368.

[5] N. J. Emery and N. S. Clayton, "The Mentality of Crows: Convergent Evolution of Intelligence in Corvids and Apes," *Science (80-.).*, vol. 306, no. 5703, pp. 1903–1907, Dec. 2004, doi: 10.1126/science.1098410.

[6] S. Ghanbari and M. Othman, "A Priority Based Job Scheduling Algorithm in Cloud Computing," *Procedia Eng.*, vol. 50, pp. 778–785, Jan. 2012, doi: 10.1016/j.proeng.2012.10.086.

[7] F. A. Omara and M. M. Arafa, "Genetic algorithms for task scheduling problem," *J. Parallel Distrib. Comput.*, vol. 70, no. 1, pp. 13–22, Jan. 2010, doi: 10.1016/j.jpdc.2009.09.009.

[8] U. Bhoi and P. N. Ramanuj, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing," *Int. J. Appl. or Innov. Eng. Manag.*, vol. 2, no. 4, pp. 259–264, 2013.

[9] S. Shirke and R. Udayakumar, "Evaluation of Crow Search Algorithm (CSA) for Optimization in Discrete Applications," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, IEEE, Apr. 2019, pp. 584–589. doi: 10.1109/ICOEI.2019.8862669.

[10] K. R. Prasanna Kumar and K. Kousalya, "Amelioration of task scheduling in cloud computing using crow search algorithm," *Neural Comput. Appl.*, vol. 32, no. 10, pp. 5901–5907, May 2020, doi: 10.1007/s00521-019-04067-2.

[11] A. M. Anter, A. E. Hassenian, and D. Oliva, "An improved fast fuzzy c-means using crow search optimization algorithm for crop identification in agricultural," *Expert Syst. Appl.*, vol. 118, pp. 340–354, Mar. 2019, doi: 10.1016/j.eswa.2018.10.009.

[12] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, and M. U. Chowdhury, "An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments," *Neural Comput. Appl.*, vol. 32, no. 6, pp. 1531–1541, Mar. 2020, doi: 10.1007/s00521-019-04119-7.

[13] A. M. Senthil Kumar and M. Venkatesan, "Multi-Objective Task Scheduling Using Hybrid Genetic-Ant Colony Optimization Algorithm in Cloud Environment," *Wirel. Pers. Commun.*, vol. 107, no. 4,

10.48047/jocaaa.2024.33.04.29

pp. 1835–1848, Aug. 2019, doi: 10.1007/s11277-019-06360-8.

[14] M. A. Elaziz, S. Xiong, K. P. N. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowledge-Based Syst.*, vol. 169, pp. 39–52, Apr. 2019, doi: 10.1016/j.knosys.2019.01.023.

[15] X. Zhao *et al.*, "Chaos enhanced grey wolf optimization wrapped ELM for diagnosis of paraquat-poisoned patients," *Comput. Biol. Chem.*, vol. 78, pp. 481–490, Feb. 2019, doi: 10.1016/j.compbiolchem.2018.11.017.

[16] H. Chen, Q. Zhang, J. Luo, Y. Xu, and X. Zhang, "An enhanced Bacterial Foraging Optimization and its application for training kernel extreme learning machine," *Appl. Soft Comput.*, vol. 86, p. 105884, Jan. 2020, doi: 10.1016/j.asoc.2019.105884.

[17] M. Wang and H. Chen, "Chaotic multi-swarm whale optimizer boosted support vector machine for medical diagnosis," *Appl. Soft Comput.*, vol. 88, p. 105946, Mar. 2020, doi: 10.1016/j.asoc.2019.105946.

[18] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, M. Tuba, and M. Zivkovic, "Task Scheduling in Cloud Computing Environment by Grey Wolf Optimizer," in *2019 27th Telecommunications Forum (TELFOR)*, IEEE, Nov. 2019, pp. 1–4. doi: 10.1109/TELFOR48224.2019.8971223.

[19] M. Jain, V. Singh, and A. Rani, "A novel nature-inspired algorithm for optimization: Squirrel search algorithm," *Swarm Evol. Comput.*, vol. 44, pp. 148–175, Feb. 2019, doi: 10.1016/j.swevo.2018.02.013.

[20] E. Cuevas, J. Gálvez, and O. Avalos, "An Enhanced Crow Search Algorithm Applied to Energy Approaches," 2020, pp. 27–49. doi: 10.1007/978-3-030-28917-1_3.

[21] S. Moghaddam, M. Bigdeli, M. Moradlou, and P. Siano, "Designing of stand-alone hybrid PV/wind/battery system using improved crow search algorithm considering reliability index," *Int. J. Energy Environ. Eng.*, vol. 10, no. 4, pp. 429–449, Dec. 2019, doi: 10.1007/s40095-019-00319-y.

[22] A. Babayan, Z. Kapelan, D. Savic, and G. Walters, "Least-Cost Design of Water Distribution Networks under Demand Uncertainty," *J. Water Resour. Plan. Manag.*, vol. 131, no. 5, pp. 375–382, Sep. 2005, doi: 10.1061/(ASCE)0733-9496(2005)131:5(375).

[23] A. M. Anter and M. Ali, "Feature selection strategy based on hybrid crow search optimization algorithm integrated with chaos theory and fuzzy c-means algorithm for medical diagnosis problems," *Soft Comput.*, vol. 24, no. 3, pp. 1565–1584, Feb. 2020, doi: 10.1007/s00500-019-03988-3.

[24] R. M. Rizk-Allah, A. E. Hassanien, and A. Slowik, "Multi-objective orthogonal opposition-based crow search algorithm for large-scale multi-objective optimization," *Neural Comput. Appl.*, vol. 32, no. 17, pp. 13715–13746, Sep. 2020, doi: 10.1007/s00521-020-04779-w.

[25] R. Akraminejad, N. Khaledian, A. Nazari, and M. Voelp, "A multi-objective crow search algorithm

10.48047/jocaaa.2024.33.04.29

for optimizing makespan and costs in scientific cloud workflows (CSAMOMC)," *Computing*, vol. 106, no. 6, pp. 1777–1793, Jun. 2024, doi: 10.1007/s00607-024-01263-4.

[26] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020, doi: 10.1109/TCYB.2020.2983860.

[27] K. R. Prasanna Kumar, K. Kousalya, S. Vishnuppriya, S. Ponni, and K. Logeswaran, "Enhanced Crow Search Algorithm for Task Scheduling in Cloud Computing," IOP Conf. Ser. Mater. Sci. Eng., vol. 1055, no. 1, p. 012102, Feb. 2021, doi: 10.1088/1757-899X/1055/1/012102.