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

Most computer operations would take more time to complete in a VM since VMs are computational elements. 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 behaviour to locate food. The crows would keep each other informed as to where the food was located until they were full. In the current investigation, it is used in a manner similar to that of crows, with the job hopping from virtual machine to VM until an ideal VM is located. The schedule is optimized until the maximum number of iterations has been reached.

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 behaviours is both unexpected and fascinating (Agarwal, 2017). 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 favourite meals and hiding places as well as the faces of its partners and other birds. It has a significant impact on the reproductive population's capacity for open and honest information sharing (Alireza, 2016).

All organisms on Earth, including humans, need food to survive. In most cases, a crow will eat

10.48047/jocaaa.2024.33.04.29

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 (Dhineshbabu, 2013). 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 behaviour 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 centre (Ebadifard, 2018). 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 (Emery, 2004). 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 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.

10.48047/jocaaa.2024.33.04.29

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 (Ghanbari, 2012).

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 (Omara, 2010).

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. Crowi is a representation of the i^{th} crow. depicts the crow's current location in the search area. The Crowi's location during iteration k is indicated by the symbol i,k (k=1,2,3. max). Aside from this, each Crowi 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 Crowi chooses to follow any of its flock members in iteration k in order to locate their

10.48047/jocaaa.2024.33.04.29

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 Crowi to steal the food supply of the $Crow_j$ is boosted in Case 1 when the Crowj is unaware of the spying activity of the Crowi, however in Case 2, the Crowj 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.

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 Crowi's location is calculated as shown in Equation.

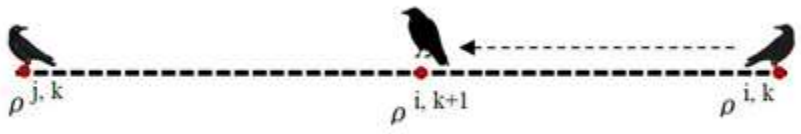


Figure 1: Position movement of $Crow_i$ based on $Crow_j$

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

Where r_i is a random value with uniform distribution, generated between 0 and1. 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

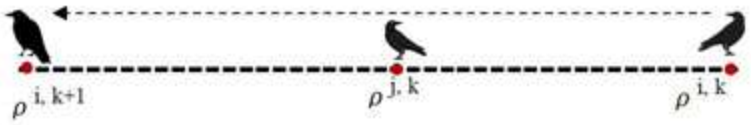


Figure 2: Random position movement of Crow $_i$

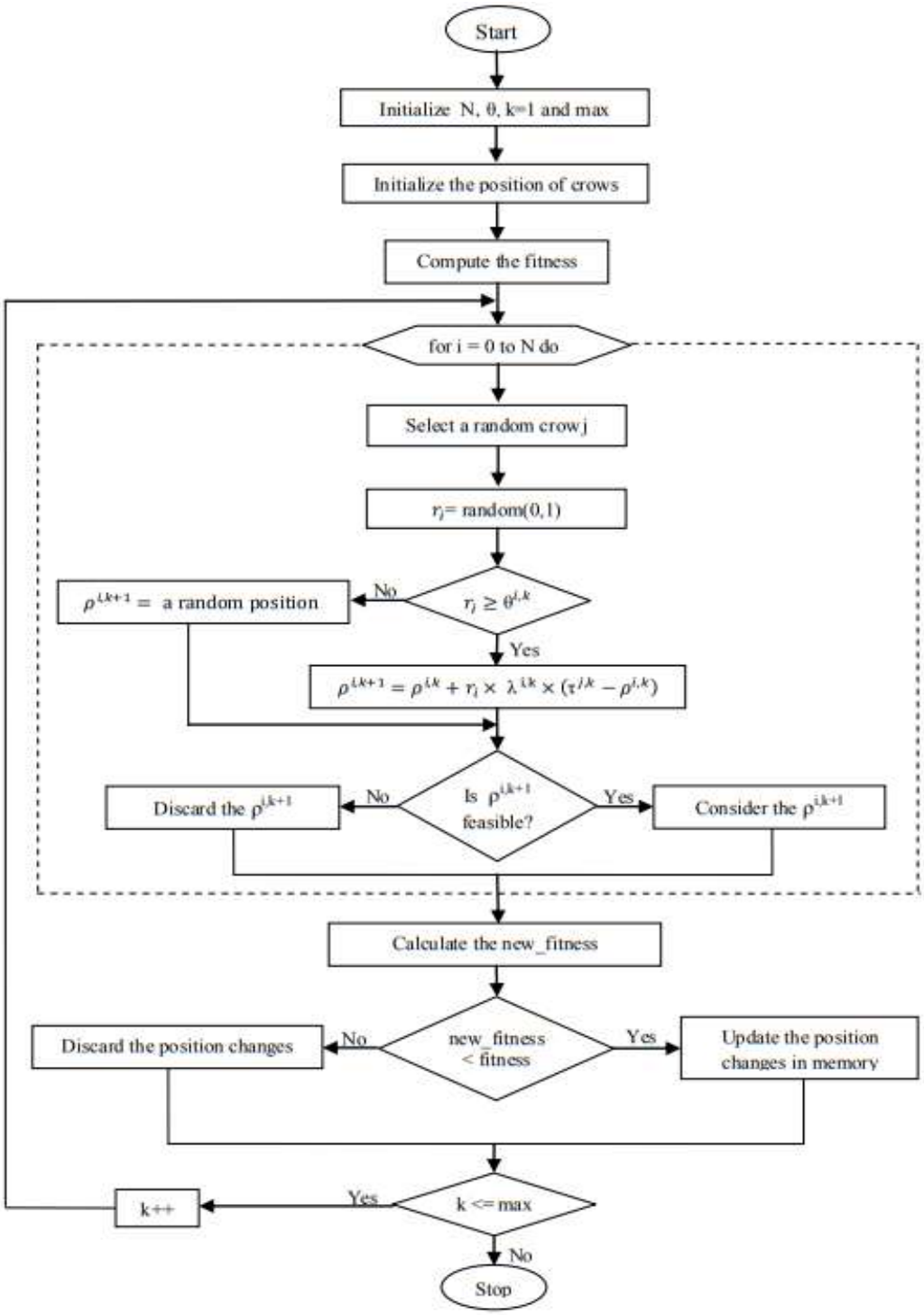
Equation shows the two situations mentioned previously.

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

Where r_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. To keep diversity and intensity in balance, the values of and are employed. Between 0 and 1, the value of increases progressively from 1.5 to 2.5.

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. Figure 3 depicts the detailed process involved in CSA.

10.48047/jocaaa.2024.33.04.29



10.48047/jocaaa.2024.33.04.29

Figure 3: Flow of CSA

Task Scheduling Using CSA

The crow's natural food collection method makes it abundantly evident that it repeats its approach in order to find a better food source. To further improve outcomes, the CSA might be used for optimization issues. It is used in cloud job scheduling to shorten makespans, lessen degrees of imbalance, and increase VM usage. Before continuing, it is assumed that each job is a crow and that the virtual machines are food sources. The VMs are dispersed around the search area at various locations. Each job would keep a VM in place while also participating in the search for a better alternative VM; once a better VM is found, the task would swap the existing VM out for the newly discovered one.

The execution time and completion time are factors taken into consideration in task scheduling, which directly affect the makespan value, when CSA begins with an initial schedule. The execution time and completion time for each job are determined using the starting schedule. Each Ti in the timetable receives the CSA. The new position in this refers to the substitute VM (VM_x). The job that Ti is supposed to do is chosen at random (T_j) from the timetable. Using the Equation, the CSA position movement is applied to the Ti depending on the values of the T_j.

$$VM_x = \begin{cases} AT_{i,k} + \left(r_i \times \lambda^{i,k} \times \left(AT_{j,k} - AT_{i,k}\right)\right) & r_j \geq \theta^{j,k} \\ a\ random\ VM & otherwise \end{cases}$$

Where AT denotes an allocation table which maintains the information of the currently allocated VM for each task in the schedule. r_i and r_j are random values generated between 0 and 1. $\theta^{j,}$ is considered as 0.05 and $\lambda^{i,}$ is increased from 1.5 to 2.5. The value of VM_x is calculated when the $r_j \geq \theta^{j,}$, Otherwise, the Ti chooses a VM at random. Calculated and compared with the task's current completion time is the task's estimated completion time in the VMx. The information of the current assigned VM will be replaced with the VMx if it offers a shorter completion time than the current allocated VM, and the task's completion time will be updated appropriately. The altered timetable is regarded as a brand-new one. Similar to this, fresh VMx is found and added in the new schedule for each job in the existing schedule.

10.48047/jocaaa.2024.33.04.29

The makespan value for the new schedule is calculated at the conclusion of the CSA algorithm and compared with that of the old schedules. The new schedule will only be taken into consideration for the next iteration if it produces a shorter makespan than the current one; otherwise, it will be rejected, and the CSA will continue with the current schedule. This CSA modifies the VM of the schedule according to its own methodology, identifying the VM that offers the shortest turnaround time. Additionally, it guarantees higher VM usage when a job is distributed across many VMs as opposed to only one. Algorithm 1, which provides a comprehensive view of the full CSA process in identifying the VM to the job, denotes each phase of CSA. With the help of CSA, the timetable is continually improved.

Makespan:

Task set is defined as T= {t1,t2,…..tm}, where m stands for how many tasks there are, and vm set is defined as vm ={vm1,vm2,…vmn} 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.

Makespan =completion time of last task – completion time of first task

Degree of imbalance:

In statistics, the variance is a measure of how far apart a collection of values are from one another. IMD assesses the effect of load imbalance by measuring the normalized variation of the average CPU, memory, and storage usage for all PMs.

$$\frac{\sum_{i=0}^{m}\left(\frac{(Avg_i - CPU_u)^2}{3} + \frac{(Avg_i - Mem_u)^2}{3} + \frac{(Avg_i - Sto_u)^2}{3}\right)}{m}$$

Where

$$Avg_i = \frac{PCPU_i^U + PMem_i^U + PSto_i^U}{3}$$

10.48047/jocaaa.2024.33.04.29

and CP Uu, Memu, Stou are the average CPU, memory, and storage usage for the Cloud data center under consideration, which may be calculated using the utilization of all PMs for the Cloud data center.

VM consumption:

Let VM=vm1,vm2,....vmn be the number of virtual machine that process m number of tasks represented as T=T1,T2,....Tm. The processing time of each job for each virtual machine is determined by the equation: Our goal is to decrease the makespan, which is shown by the symbol CPmn.

$P_{mn}=C_m/PS_n$

Where Pmn is the virtual machine's processing time and Cm is the task's computational complexity, PSn is the virtual machine's processing speed.

The VM's total capacity is determined as follows:

$$C= \sum_{m=1}^{n} Cm$$

Algorithm 1: CSA for Task Scheduling

Input:

N, M, max

Output:

Schedule

CSA_Algorithm:

Begin

Initialize the values for AP, FL and count=1

Generate the initial schedule and store allocation details in AT

Calculate CT for tasks based on schedule

10.48047/jocaaa.2024.33.04.29

Makespan = Identify_makespan(N, CT)

While (k≤ max) do

For i=1:N do

T_j= random(0,N)r

$_i$ = random(0,1)

r_j= random(0,1)

If $r_j \geq AP_{j,k}$ then

$$x = AT_{i,k} + (r_i \times FL_{i,k} \times (AT_{j,k} - AT_{i,k}))$$

else

x= random(0,M)

End If

$TCT_i = FT_x + ETC_{i,x}$

If $TCT_i < CT_i$ then

$TAT_{i,k} = x$

else

End If

End For

$TAT_{i,k} = AT_{i,k}$ $TCT_i = CT_i$

TM= Identify_makespan(N, TCT)

If TM < Makespan then

Update the TAT values in AT and TCT values in CT

10.48047/jocaaa.2024.33.04.29

Makespan= TM

End If

k=k+1

End While

Output the schedule

Calculate the degree of imbalance and VM utilization

End

Algorithm 2: Identify Makespan Algorithm

Input:

N, CT

Output:

Makespan

Identify_makespan:

Begin

Makespan = CT_1

For i = 2 to N do

If Makespan < CT_i then

Makespan = CT_i

End If

End For

End

10.48047/jocaaa.2024.33.04.29

The CSA is applied to the schedule for a predetermined number of iterations, which improves the schedule with minimal makespan, but in certain circumstances, the new schedule's makespan is larger than the existing schedule's makespan since the tasks and VM are randomly chosen when $r_j < \theta^{i,k}$ in CSA. By making changes to the CSA's random selection process, performance may be further improved.

ECSA for Task Scheduling

Because T_j is chosen at random for each iteration in CSA, the outcome might be impacted or improved. Even though there are many crows in the flock, choosing the right partner to pursue is crucial to the CSA process since it improves the hit ratio of the $Crow_i$. Similar to this, choosing a more beneficial work will enhance the schedule even more. Therefore, choosing the right Tj is crucial to enhancing the CSA's tasking scheduling performance. It may be carried out in a variety of ways by looking at the current timetable in a manner like to the crow's snooping. The execution time and completion time are the two factors that are most directly related to each job in the task scheduling issue under consideration. The properties of the particular VM and the task's duration alone determine how long it takes to complete the job. Additionally, it could not take the VM's workload or current availability into account. Therefore, choosing a VM based on execution time may not result in greater performance.

Another important criterion is the completion time, which is calculated by summing the task's execution time with the time the VM currently has available. The task's new completion time would be longer than its current completion time if the VM is already assigned to additional tasks. Therefore, the completion time suggests the VM's present load in an indirect manner. Choosing a task Tj that completes in a significant amount less time than Ti produces better results than the random selection approach because if a VM completes a work more quickly, it may also complete other tasks that are comparable in speed. The CSA process is improved in this manner, and the new form is known as ECSA. Algorithm 3 outlines the processes needed to identify the job that will take the least amount of time to complete. ECSA shortens the schedule's makespan and increases the likelihood that Ti will get a better virtual machine.

Algorithm 3 Task Search Algorithm

Input:

10.48047/jocaaa.2024.33.04.29

CT_i, CT

Output:

Task id

Search_min_task:

Begin

For j = 1 to N do

If $CT_j <$ CT_i then

return j

End If

End For

End

Similarly, when the r_j value is lesser than $\theta^{i,}$, Because it is comparable to how the crow chooses its random spot, a random VM is chosen. Since it is unknown what state the randomly chosen VM is in, there is room for various inquiries about performance and the result of the schedule. For instance, if the randomly chosen VM is already fully loaded, this will further impair its performance and lengthen its total makespan. Therefore, it is preferable to choose a VM depending on the present workload as well as the one that is easily executable. The workload among the VMs would be balanced using this method of VM selection. Every VM has a current load value, which is computed, and an average load, Avg load, which is also calculated. The VM selection method chooses an appropriate VM with a load that is lower than the average load and has the least amount of spare time. This is an addition to the CSA algorithm that further shortens the makespan and distributes the burden across the VMs. procedures for the VM selection method listed in Algorithm 4.

Algorithm 4 VM Selection Algorithm

Input:

10.48047/jocaaa.2024.33.04.29

M, FT, LD

Output:

Min_VM

Select_VM_Algorithm:

Begin

For j=1 to M do

Total=Total+LD_j

End For

Avg_load=Total/M

Min_FT=FT_1

Min_VM=1

For j=2 to M do

If ((FT_j< Min_FT) && (LD_j< Avg_load)) then

Min_VM=j

Min_FT=FT_j

End If

End For

return Min_VM

End

Result and Discussions of CSA and ECSA

The CloudSim simulator is used to test the Max-Min algorithm, GA, CSA, and ECSA. With 16

10.48047/jocaaa.2024.33.04.29

virtual machines and jobs ranging from 100 to 500, the simulation environment is set up. The flight duration is raised from 1.5 to 2.5, and the tuning parameters of awareness probability are set at 0.05.

The performance of the CSA, ECSA, Max-Min, and GA algorithms based on makespan, with a number of jobs raised from 100 to 500 progressively, is shown in Figure 4. When compared to Max-Min and GA, it is clear that the makespan value of the CSA and ECSA is well reduced. The CSA results in improvements of 4% to 6% over GA and 7% to 10% over the Max-Min algorithm. Similarly, ECSA results in improvements of 8% to 13% when compared to GA, 11% to 16% when compared to Max-Min algorithm and 4% to 7% when compared to CSA based on makespan values.

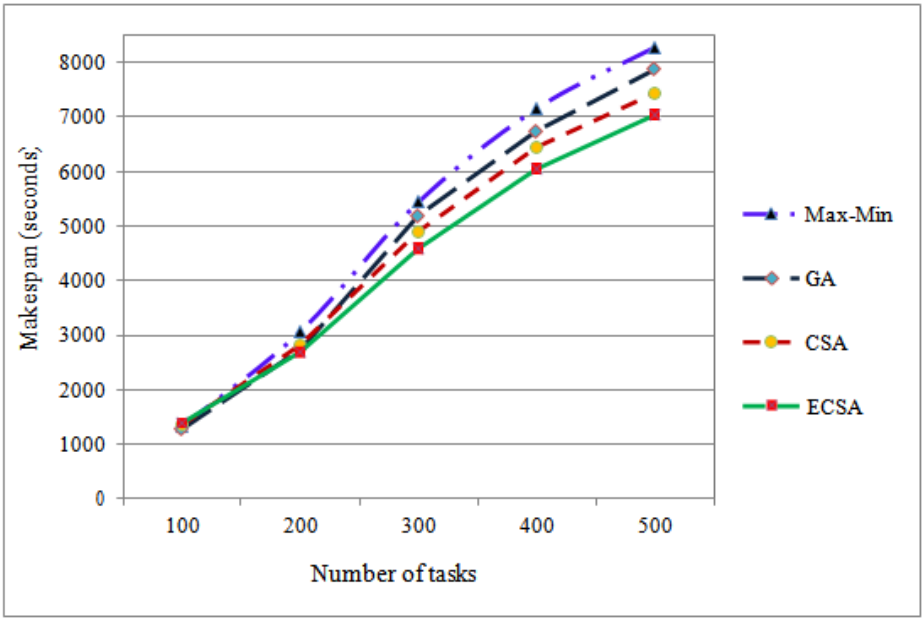


Figure 4: Max-Min, GA, CSA, ECSA span comparison

Additionally, 16 VMs and 500 jobs are taken into consideration for the comparison of makespan values between the CSA and ECSA while adjusting the iterations from 10 to 100. Figure 5 displays the CSA and ECSA's progress. When the number of iterations is increased, the makespan of the schedule progressively grows shorter, but the difference becomes extremely little after the count exceeds 50. It is obvious that having an iteration count between 50 and 100 would result in the makespan value being as short as possible.

10.48047/jocaaa.2024.33.04.29

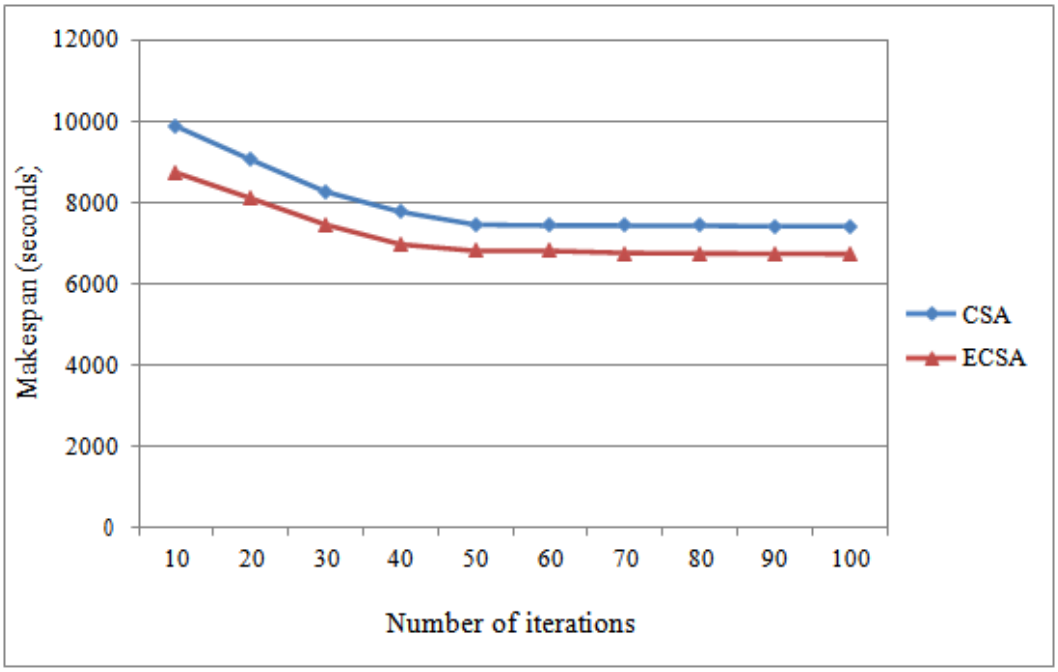


Figure 5: Makespan-based CSA and ECSA iteration performance

Due to advancements made in the VM selection process, CSA and ECSA, as compared to GA and Max-Min algorithms, lower the degree of imbalance even when the number of tasks is steadily increased, as shown in Figure 6. The ECSA's VM selection algorithm takes into account the job's completion time, current load, and free time to make sure the work is assigned to the VM with the least amount of burden. In comparison to even CSA, the ECSA achieves a low overall degree of imbalance, which ensures that it evenly distributes the load across the VMs.

10.48047/jocaaa.2024.33.04.29

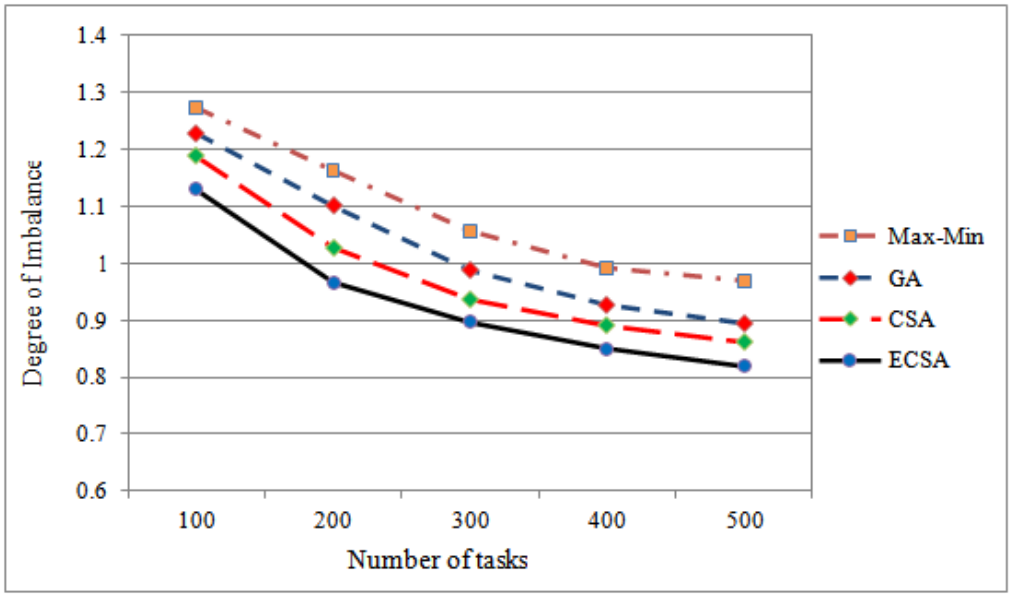


Figure 6: Max-Min, GA, CSA, ECSA imbalance comparison

The VM utilization is taken into account as another parameter. TheCSA and ECSA utilize the VM in a better way while comparing with GA and Max-Min as shown in Figure 7. The CSA algorithm allocates the task to VM which gives minimum completion time. So, it mostly distributes the task to different VMs and performs better than GA and Max-Min. Due to the improvement made on the ECSA in the task selection and VM selection process, the VM utilization is further improved. The ECSA utilizes the VM very effectively and a high level of utilization is reached.

10.48047/jocaaa.2024.33.04.29

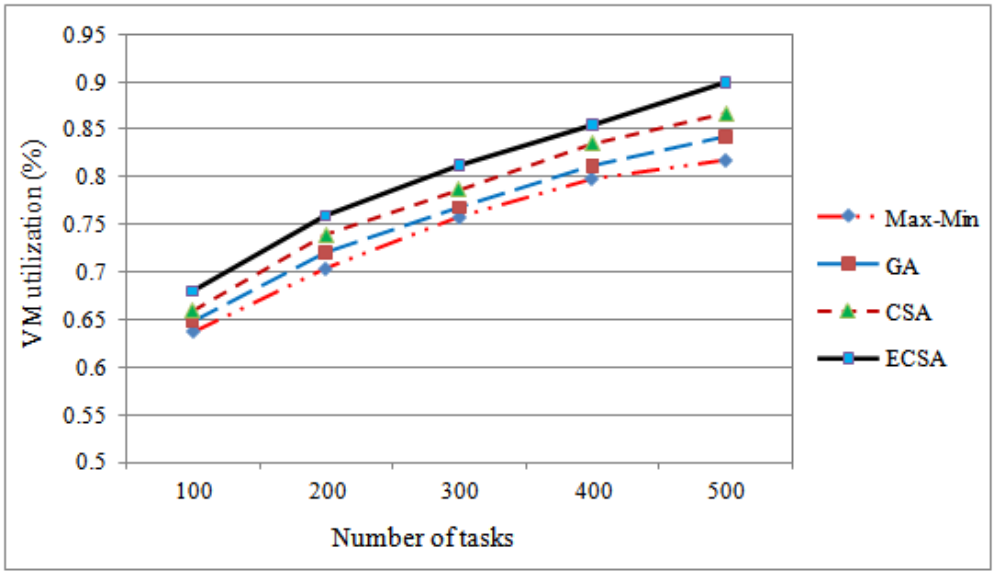


Figure 7: Max-Min, GA, CSA, and ECSA VM usage comparison

Conclusions

The CSA and ECSA were defined in the research for cloud task scheduling. CSA and ECSA are promising for task scheduling in the cloud because they are resilient simple algorithms with two movable parameters: flight time and awareness probability. The random selection of the CSA algorithm in ECSA is directly influenced by the task selection and VM selection methods. Based on several benchmarks, including makespan, degree of imbalance, and VM consumption, the outcomes of CSA and ECSA have been compared with GA and Max-Min. Based on the set of benchmark values, it is shown that CSA and ECSA beat GA and Max-Min algorithms. From the findings, it can be observed that CSA and ECSA function well and determine the best plan for task scheduling in the cloud.

References

1. Agarwal, M & Srivastava, GMS 2017, 'A Cuckoo Search AlgorithmBased Task Scheduling in Cloud Computing', Advances in Computer and Computational Sciences. Advances in Intelligent Systems and Computing, vol. 554, pp. 293-299
2. Alireza Askarzadeh 2016, 'A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm', Computers & Structures, vol. 169, pp. 1-12

10.48047/jocaaa.2024.33.04.29

3. Bhoi, U & Ramanuj, PN 2013, ‘Enhanced max-min task scheduling algorithm in cloud computing’, International Journal of Application or Innovation in Engineering and Management, vol. 2, no. 4, pp. 259-264.
4. Dhineshbabu, LD & Venkatakrishna, P 2013, 'Honey bee behavior inspired load balancing of tasks in cloud computing environments', Journal of Applied Soft Computing, vol. 13, no. 5, pp. 2292-2303.
5. Ebadifard, F & Babamir, SM 2018, ‘A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment’, Concurrency and Computation: Practice and Experience, vol. 30, no. 12, pp. 1-16.
6. Emery, NJ & Clayton, NS 2004, 'The mentality of crows: convergent evolution of intelligence in corvids and apes', science, vol.306, no.5703, pp. 1903-1907.
7. Foster, I, Zhao, Y, Raicu, I & Lu, S 2008, 'Cloud computing and grid computing 360-degree compared', Proceedings of Grid Computing Environments Workshop, Austin.
8. Gandomi, AH, Yang, XS & Alavi, AH 2013, 'Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems', Engineering with Computers, vol. 29, no. 1, pp.17–35.
9. Ghanbari, S & Othman, M 2012, ‘A priority based job scheduling algorithm in cloud computing’, in Advances Science and Contemporary Engineering : Proceedings of the International Conference, vol. 50, pp. 778-785.
10. Jason, Brownlee 2012, Clever Algorithms: Nature-Inspired Programming Recipes, LuLu Press Inc, United States.
11. Min-Yuan, Cheng & Doddy, Prayogo 2014, 'Symbiotic organisms search: A new metaheuristic optimization algorithm', Computers & Structures, vol. 139, pp. 98–112.
12. Omara, FA & Arafa, MM 2010, ‘Genetic algorithms for task scheduling problem’, Journal of Parallel and Distributed Computing, vol. 70, no. 1, pp. 13-22.
13. Senthil Kumar A M ,Venkatesan M , Multi-Objective Task Scheduling Using Hybrid Genetic-Ant Colony Optimization Algorithm in Cloud Environment, Wireless Personal Communications, 2019
14. Elaziz M A,Xiong S,Jayasena K P N and Li L 2019 Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolutionKnowledge-Based System vol 169p 39–52

10.48047/jocaaa.2024.33.04.29

15. Xiao X and Li Z 2019 Chemical Reaction Multi-Objective Optimization for Cloud Task DAG Scheduling IEEE Access vol 7 p 102598–102605
16. Singh H 2020Crow – penguin optimizer for multiobjective task scheduling strategy in cloud computingp1–18
17. Bacanin N,Bezdan T,Tuba E,Strumberger I, Tuba M and Zivkovic M 2019 Task Scheduling in Cloud Computing Environment by Grey Wolf Optimizer
18. Zhou Z, Li F, Zhu H,Xie H,Abawajy J H and Chowdhury,M U 2019An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments Neural Computing Applications, vol 7
19. Monisha R,Mrinalini R, Britto M N, Ramakrishnan R and Rajinikanth V2019 Smart Intelligent Computing and Applications vol 104
20. Shirke S and Udayakumar R 2019 Evaluation of Crow Search Algorithm (CSA) for optimization in discrete applicationsProceedings of the International Conferenceon Trends in Electronics and Informatics(ICOEI)p 584–589
21. Prasanna Kumar K R and Kousalya K 2020 Amelioration of task scheduling in cloud computing using crow search algorithm Neural Computing Applications vol 32p 5901–5907
22. Wang, M., & Chen, H. (2020). Chaotic multi-swarm whale optimizer boosted support vector machine for medical diagnosis. *Applied Soft Computing*, *88*, 105946.
23. Zhao, X., Zhang, X., Cai, Z., Tian, X., Wang, X., Huang, Y., ... & Hu, L. (2019). Chaos enhanced grey wolf optimization wrapped ELM for diagnosis of paraquat-poisoned patients. *Computational biology and chemistry*, *78*, 481-490.
24. Chen, H., Zhang, Q., Luo, J., Xu, Y., & Zhang, X. (2020). An enhanced bacterial foraging optimization and its application for training kernel extreme learning machine. *Applied Soft Computing*, *86*, 105884.
25. Sun, Y., Xue, B., Zhang, M., Yen, G. G., & Lv, J. (2020). Automatically designing CNN architectures using the genetic algorithm for image classification. *IEEE transactions on cybernetics*, *50*(9), 3840-3854.
26. Yu, H., Zhao, N., Wang, P., Chen, H., & Li, C. (2020). Chaos-enhanced synchronized bat optimizer. *Applied Mathematical Modelling*, *77*, 1201-1215.

10.48047/jocaaa.2024.33.04.29

27. Jain, M., Singh, V., & Rani, A. (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm and evolutionary computation*, *44*, 148-175.
28. Hashim, F. A., Houssein, E. H., Mabrouk, M. S., Al-Atabany, W., & Mirjalili, S. (2019). Henry gas solubility optimization: A novel physics-based algorithm. *Future Generation Computer Systems*, *101*, 646-667.
29. Abdallh, G. Y., & Algamal, Z. Y. (2020). A QSAR classification model of skin sensitization potential based on improving binary crow search algorithm. *Electronic Journal of Applied Statistical Analysis*, *13*(1), 86-95.
30. Alomoush, W., Alrosan, A., Alomari, Y. M., Alomoush, A. A., Almomani, A., & Alamri, H. S. (2022). Fully automatic grayscale image segmentation based fuzzy C-means with firefly mate algorithm. *Journal of Ambient Intelligence and Humanized Computing*, *13*(9), 4519-4541.
31. Anter, A. M., Hassenian, A. E., & Oliva, D. (2019). An improved fast fuzzy c-means using crow search optimization algorithm for crop identification in agricultural. *Expert Systems with Applications*, *118*, 340-354.
32. Wang, G. G., Gandomi, A. H., Alavi, A. H., & Gong, D. (2019). A comprehensive review of krill herd algorithm: variants, hybrids and applications. *Artificial Intelligence Review*, *51*(1), 119-148.
33. Wu, H., Wu, P., Xu, K., & Li, F. (2020). Finite element model updating using crow search algorithm with Levy flight. *International Journal for Numerical Methods in Engineering*, *121*(13), 2916-2928.
34. Cuevas, E., Gálvez, J., & Avalos, O. (2020). An enhanced crow search algorithm applied to energy approaches. In *Recent Metaheuristics Algorithms for Parameter Identification* (pp. 27-49). Springer, Cham.
35. Moghaddam, S., Bigdeli, M., Moradlou, M., & Siano, P. (2019). Designing of stand-alone hybrid PV/wind/battery system using improved crow search algorithm considering reliability index. *International Journal of Energy and Environmental Engineering*, *10*(4), 429-449.
36. Fallah, H., Kisi, O., Kim, S., & Rezaie-Balf, M. (2019). A new optimization approach for the least-cost design of water distribution networks: improved crow search algorithm. *Water Resources Management*, *33*(10), 3595-3613.

10.48047/jocaaa.2024.33.04.29

37. Anter, A. M., & Ali, M. (2020). Feature selection strategy based on hybrid crow search optimization algorithm integrated with chaos theory and fuzzy c-means algorithm for medical diagnosis problems. *Soft Computing*, *24*(3), 1565-1584.
38. Rizk-Allah, R. M., Hassanien, A. E., & Slowik, A. (2020). Multi-objective orthogonal opposition-based crow search algorithm for large-scale multi-objective optimization. *Neural Computing and Applications*, *32*(17), 13715-13746.
39. Spea, S. R. (2020). Solving practical economic load dispatch problem using crow search algorithm. *International Journal of Electrical & Computer Engineering (2088-8708)*, *10*(4).
40. Prasanna Kumar, K. R., & Kousalya, K. (2020). Amelioration of task scheduling in cloud computing using crow search algorithm. *Neural Computing and Applications*, *32*(10), 5901-5907.
41. Rathore, A., & Mishra, S. (2024). Improved cloud task scheduling using an adaptive crow search algorithm. Journal of Cloud Computing and Optimization, 15(2), 112–125.
42. Patel, R. K., Singh, D., & Verma, L. (2024).A multi-objective crow search algorithm for efficient workflow scheduling in heterogeneous cloud environments. International Journal of Distributed Cloud Systems, 9(4), 287–299.