ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20230913000935
Original scientific paper

An Enhanced Trust Scheduling Algorithm for Medical Applications in a Heterogeneous Cloud Computing Environment

Ganapriya K*, Poobalan A, Gopinath S, Vedha Vinodha D

Abstract: This paper aims to present and deploy an improved task scheduling algorithm for the allocation of user tasks across multiple computing resources. The primary goal of this algorithm is to minimize both execution time and costs while simultaneously enhancing resource utilization within the context of medical applications. Virtual machine scheduling in a heterogeneous cloud environment needs significant attention with the increase in the usage of cloud resources by end users and enterprises. It is one of the significant parameters that affects cloud data centers. The resources requested by every user vary in their configuration. Finding a suitable virtual machine for each process is dynamically a time-consuming process. Virtual machines are classified based on resources such as memory and processing units. Upon the arrival of a request with specific requirements, it can be effortlessly mapped to a corresponding virtual machine. This process is followed by a bilateral method encompassing queuing and scheduling. Queues are formed for requests with different requirements, which are followed by a scheduling algorithm that allocates VMs based on the minimum remaining resources in the resource pool. A scheduling mechanism has been designed to solve the problem of starvation that occurs with the Min-Min fit scheduling policy. The average turnaround time and waiting times are observed to be significantly reduced, which has an impact on the performance of the data center for medical applications. Using the CloudSim Plus tool, the experimental outcomes demonstrated that the proposed approach exhibited remarkable superiority over competing methods in relation to metrics such as average waiting time, turnaround time, and response time. This advantage was observed when compared to multiple algorithms that were examined during the study.

Keywords: cloud computing; medical applications; quality of service; scheduling; virtual machine

1 INTRODUCTION

Through the application of virtualization, the cloud computing paradigm offers customizable, scalable services on-demand, tailored to the specific requirements of its users. These systems automatically allocate resources to guests based on the Service Level Agreement (SLA) agreed upon between the cloud provider and customer. An effective scheduling method is required to allocate these resources accurately based on the type of demands placed onto the cloud platform. In the cloud paradigm, an inadequate and inefficient work scheduler reduces service quality by breaching SLA requirements, resulting in a loss of trust in the cloud provider. The current circumstances necessitate the development of a task scheduling algorithm that incorporates task dependencies, integrates the power cost aspect of virtual machines (VMs), and optimizes task scheduling to minimize the overall time required for completion (makespan). Simultaneously, the objective is to enhance resource utilization by reducing underutilization (vacuity), improving the success rate of task execution, and increasing the efficiency of task reversals [1].

Virtual machine scheduling is one of the critical factors that have a considerable impact on the performance of the cloud data center. As a result, an efficient trust-aware task scheduling algorithm was developed, known as TAFFA (Trust-Apprehensive Firefly Optimization). This algorithm successfully achieves the scheduling of task precedence and virtual machines while considering their electricity unit costs. Furthermore, a correlation has been identified between the makespan and trust aspects, rooted in the framework of Service Level Agreement (SLA) parameters. The assessment of trust based on the SLA is interconnected, encompassing elements such as the success rate of the virtual resource, the degree of underutilization of VMs (vacuity), and the efficiency of task reversals [2].

In this investigation, a deadline constraint is incorporated to guide the allocation of tasks to virtual repositories following the completion of ongoing tasks. Comprehensive simulations were conducted using CloudSim. Synthetic workload distributions, including linear, normal, left-skewed, and right-skewed distributions, were generated, alongside real-time worklogs obtained from HPC2N and NASA.

The recommended TAFFA algorithm was compared to the PSO, ACO, and GA algorithms. TAFFA outperformed these methods in key aspects such as resource utilization, success rate, and efficiency, all while reducing the makespan, according to the simulation findings. Cloud computing is a game-changing technology that provides on-demand access to critical resources in storage, processing, and networking. This shift facilitates elastic, pay-as-you-go services delivered via the internet. The effectiveness of cloud data centers relies on a multitude of influencing factors [3].

Response and waiting times are the decisive factors in offering a high quality of service to cloud consumers. The virtual machine scheduling technique is used to determine these performance characteristics. This also addresses the hunger [4, 5] issue caused by the Min-Min Best fit method and reduces average waiting time and turnaround times. Fig. 1 shows the various services of the cloud-based process in a heterogeneous network.

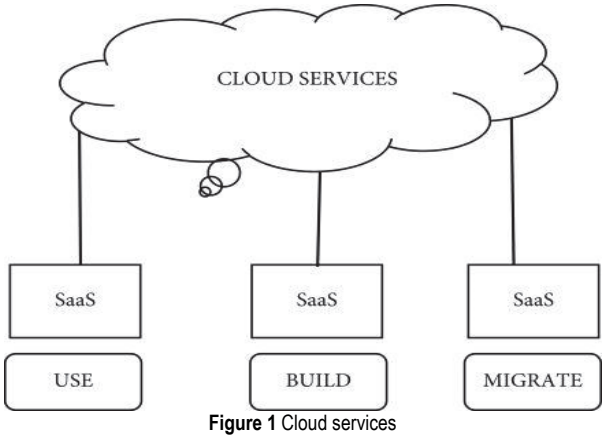

Figure 1 Cloud services

The following section contains state-of-the-art works that manage task scheduling and virtual machine scheduling. The third section provides the existing practice method, and the fourth section provides the recommended solution. The fifth section provides the experimental data, while the last section describes the conclusion and the scope of future research. Fig. 2 shows the overall architecture of a cloud-based process or task scheduling system.

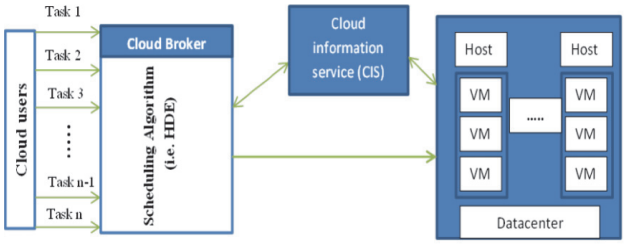

Figure 2 System architecture

2 LITERATURE SURVEY

Task scheduling stands as a critical concern and represents a prominent challenge in the context of Cloud Computing. This is due to the service provider's responsibility to cater to numerous user applications originating from diverse locations and timeframes. Consequently, the imperative need to efficiently and swiftly schedule these tasks for execution within the Cloud computing environment is paramount. To address this, a multitude of meta-heuristic task scheduling algorithms have emerged, leveraging methodologies like PSO and GA [6-10]. Given that the suggested task scheduling approach relies on the PSO algorithm, the forthcoming section will delve into the pertinent research related to PSO. The existing system considered is the scheduling policy that combines the shortest job first and the Min-Min fit scheduling. In this model, the system chooses a sequence that always minimizes the remaining resource. For example, consider the resource pool given below:

Memory = 30 GB
CPU = 30
Storage = 400 GB

VM instances are classified into types as shown below in Tab. 1.

Table 1 Different types of VM instances

Instance Type	Memory	CPU	Storage
Type 1	15GB	8EC2 unit	1690 GB
Type 2	17.1 GB	6.5 EC2unit	420GB
Type 3	7GB	10 EC2unit	1690GB

Consequently, the shortest job first algorithm partitions the requests according to their corresponding virtual machine requirements, followed by the subsequent application of the Min-Min fit strategy [11-17]. For the following are the potential combinations of virtual machines that may be performed given the resource pool and the VM instance installation.

[[0, 0, 0], [1, 0, 0], [0, 1, 0], [0, 0, 1],
[1, 0, 1], [0, 1, 1], [2, 0, 0]]

Among these combinations that best fit those results in minimal balance resources is [2 0 0].

To propose and develop an "Iterated Greedy Algorithm" tailored to tackle the specific scheduling problem mentioned the paper aims to introduce a novel or enhanced algorithm for solving this problem efficiently. The paper may seek to optimize various scheduling criteria, such as minimizing makespan (total job completion time), reducing production time, or improving resource utilization, while considering re-entrant and group processing [18, 19].

It specifies that 2 instances of type1 virtual machine can be executed, but if it supposes there are continuous type 1 request, then the type 1 and type 2 instances will be pushed to a starvation state. This is the problem considered and solved [20-25].

We analyzed the behaviour of the algorithm by evaluating the three datasets used, which contain jobs with a distinct burst time order but an arrival time equal to zero, to find the key gaps and shortages with the suggested methods stated in the literature review. Samples from such a collection are with numerical values according to their qualities.

3 PROPOSED SYSTEM

The main objective of the proposed system is to create a model that avoids the issue that occurs with employing the Min-Min fit technique. For avoiding the issue discussed in the previous section the following steps are used. A combination that supports execution of multiple instances of the same virtual machine is avoided. Here in the example provided [2 0 0] it is avoided. In order to avoid starvation, the remaining combinations are considered. The initial combination from which the process should start according to min-min fit is [1, 0, 1] But it is not chosen because it would make type2 instances wait for a long period. Instead of that [0 1 1] is chosen initially and when again a decision is to be made. The min fit is followed. The rationale behind this selection will be elucidated through an example in the subsequent sections. Fig. 3 shows the concept about the task scheduling approach.

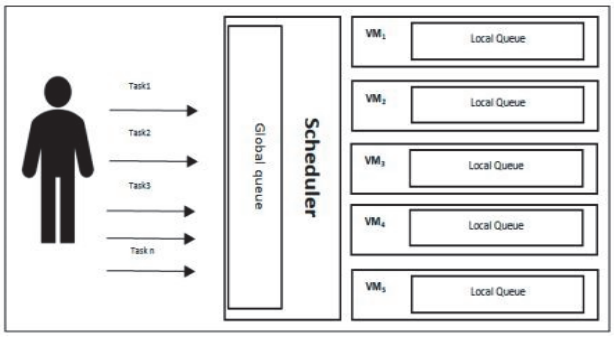

Figure 3 Task scheduling approach

3.1 System Architecture

Assume there is a collection of n jobs, each with a fixed execution time on a given processing unit. These tasks are designed to be carried out on m computational resources. It should be noticed that m is smaller than the number of tasks (i.e., *n*), meaning that tasks cannot be spread across various resources, thereby preventing task migration. The primary goal of task scheduling is to

minimize work durations and related costs while improving resource usage. Fig. 4 displays the proposed system's general design.

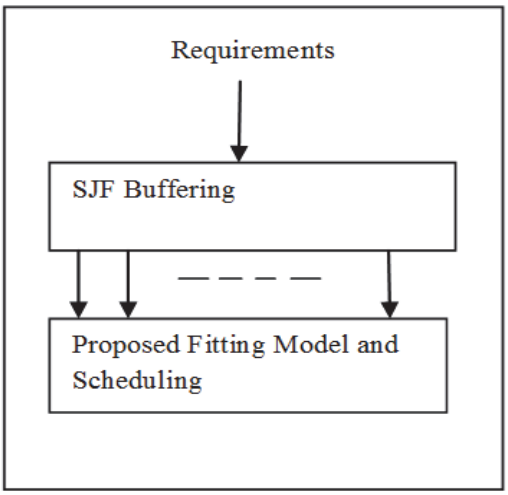

Figure 4 System architecture

The system receives the instance needs, which are organized in distinct queues based on the kind and shortest job first policy. The processes in the queue are extracted based on the best solution given.

The employed scheduling strategy in this context entails a fusion of the shortest job first principle and the adapted Min-Min fit approach. They are described below.

i. Min-Min best fit scheduler which acts as the dispatcher of the processes follows the following steps.

ii. With the given resource pool and the requirements from the users, the policy will find the different combination of the virtual machine instances.

iii. Among the combinations the one which best fit is identified.

Best fit is the one which has the least remaining resource in the resource pool. The following example illustrates this. For the given resource pool and virtual machine instances the remaining resources for different combinations are as follows.

For [2 0 0] the remaining resource in the resource pool is:

Remaining resource = [0 14 620] [2 0 0] specifies that 2 instances of the type 1 virtual machine instances can be executed at the same time.

The other combinations that can be considered are:

[1 0 1] and [0 1 1]

Others can be neglected for finding the best fit since only one instance of any one type can be executed at a particular time.

For [1 0 1], one instance of type 1 and one instance of type 3 can be executed at the same time. The remaining resource in the resource pool for this combination is

Remaining resource = [7.9 2 620]

For [0 1 1], one instance of type 2 and one instance of type 3 can be executed at the same time. The remaining resource in the resource pool for this combination is

Remaining resource = [5.9 21.5 1890]

Among these three combinations, obviously the best fit is [2 0 0].

Fig. 5 shows the overall architecture of the proposed scheduling flow diagram.

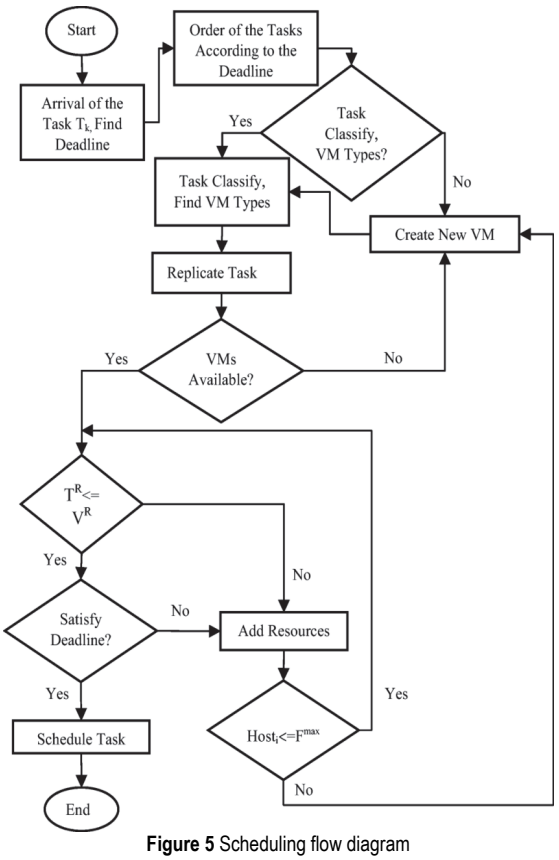

Figure 5 Scheduling flow diagram

3.2 Starvation Problem

But the problem that arises with this combination is the following: when there is a continuous flow of the type 1 jobs, the type 2 jobs and type 3 jobs go to the starvation state.

3.3 Proposed Solution

In order to avoid the starvation problem, the following is done. Fig. 6 shows the scheduling flow diagram for the proposed system.

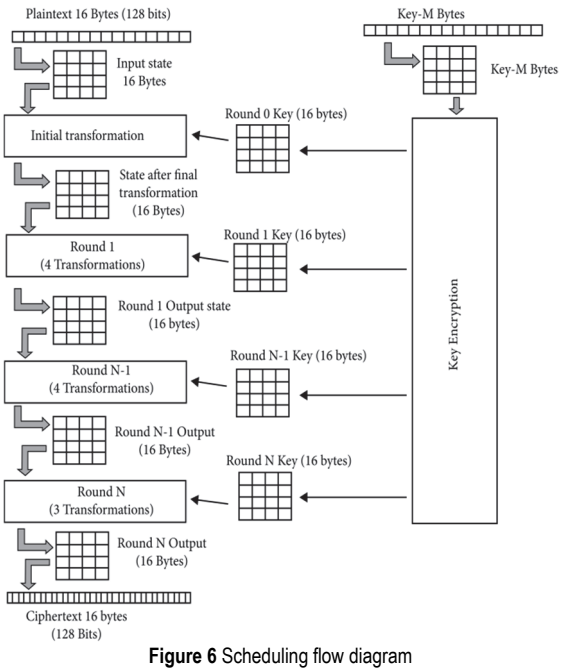

Figure 6 Scheduling flow diagram

The combination which provides option for execution of multiple instances of the same type of virtual machines is omitted, hence [2 0 0] is omitted. The remaining combinations available with multiple instance provision are [1 0 1] and [0 1 1].

When the Min-Min best fit approach is applied, the best fit is [1 0 1]. It has been observed from the execution that this model is selected for initialization, the waiting time is long for type 2 jobs and there arises a situation where only single instances of the virtual machines are executed and the turnaround time is obviously high.

Now, the requirement is to find the victim; victim here refers to the type of virtual machine which would have a long waiting time, or the risk of getting in to the starvation state. This is the one which does not have a combination with the type of instance which has a combination of executing multiple instances. For example here the combination that makes multiple instances of same machine executing is [2 0 0], the corresponding virtual machine type is type1.

Observe the other combination which allows two different virtual machines to be executed in parallel.

[[[1, 0, 1],[0, 1, 1]]]

Among these two type 3 VM has the combination with the type 1 but the type 2 does not have. It is found to be the victim. So, the initial combination that has to be started with is the victim. The following is the trace of the scheduling. Each entity v_{i-j}, t where V_i represents the virtual machine instance, j represents the nth instance of the type v virtual machine, and t represents the time space.

$$\begin{array}{lll} (v_{1-3,4}) & (V_{1-2,4}) & (v_{1-1,2}) \\ (V_{2-2,1}) & (V_{2-1,1}) & \\ (V_{3-2,3}) & (v_{3-1,1}) & \end{array}$$

The above is the requirement arranged in the different queues as per the type of the virtual machines following the SJF. The best fit option as per the proposed model is [0 1 1].

$$\begin{array}{l} (V_{2-1,1}) \\ (v_{3-1,1}) \end{array}$$

Since a process is completed, again a decision has to be made to choose the best fit. Again, the best fit here is [0 1 1].

$$\begin{array}{ll} (V_{2-1,1}) & (V_{2-2,1}) \\ (v_{3-1,1}) & (v_{3-2,3}) \end{array}$$

The next best fit would be [1 0 1].

$$\begin{array}{llll} (V_{2-1,1}) & (V_{2-2,1}) & (v_{1-1,2}) & (v_{1-1,2}) \\ (v_{3-1,1}) & (v_{3-2,3}) & (v_{3-2,3}) & (v_{3-2,3}) \end{array}$$

Now again a decision is to be made, but there are no type 2 and type 3 instances hence the type 1 instances are allocated continuously. The total time spaces required are 11 which is higher than the time spaces which consider the [2 0 0] combination.

However, a substantial reduction is observed in the average waiting time and average turnaround time. The results obtained are discussed in the next section.

4 EXPERIMENTAL RESULTS

This section provides a detailed description of extensive simulations conducted using the CloudSim simulator, which effectively replicates the entire cloud environment through Java programming tools. In the course of this research, two separate workloads were employed for these simulations, encompassing randomized workloads that showcased various distributions, such as uniform, normal, left-skewed, and right-skewed distributions. To evaluate the system's performance, we applied a methodology that necessitated the utilization of simulation parameters as outlined in Tab. 2.

Table 2 Simulation parameters

Name	Quantity
Number of Tasks	10-100
Length	500,000
Memory	32 GB
Storage capacity	1 TB
Network bandwidth	100 Mbps
No. of Virtual Machines	5
Memory capacity	4 GB
Bandwidth	40 Mbps
Processing elements	2000 MIPS

The system is tested with the process which requires the following execution time shown in Tab. 3.

Table 3 Process with execution time

S. No.	Process	Time Required
1	P1	2
2	P2	4
3	P3	4
4	P4	1
5	P5	1
6	P6	1
7	P7	3

The following Fig. 7 represents the Average waiting time obtained with the shortest job first algorithm executed in a single virtual machine, shortest job first algorithm along with min-min best fit algorithm and shortest job first algorithm with the proposed model.

SJF - Shortest Job First Algorithm

SJF(MQ) - Min-Min-Shortest job first Algorithm with Multiple Queues and Min-Min Best Fit Algorithm

SJF(MQ) - Proposed - Shortest job first Algorithm with Multiple Queues and proposed Algorithm.

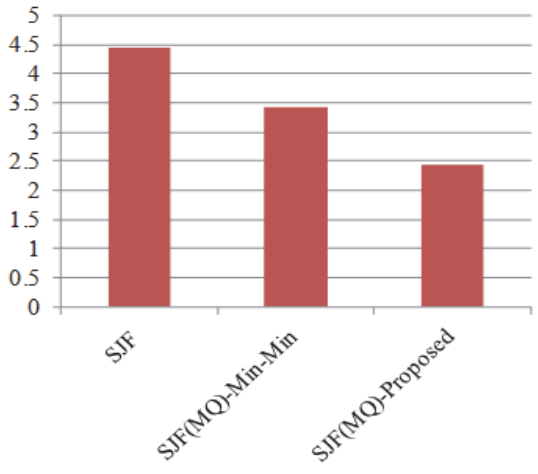

Figure 7 Comparison of the average waiting time

It can be clearly inferred from the results that the average waiting time of the proposed model is reduced when compared to the other two models.

The following Fig. 8 represents the comparison of the waiting time for the individual processes of the three methods.

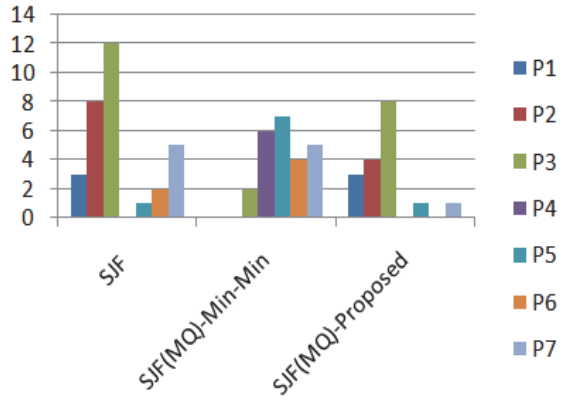

Figure 8 Waiting time of all the processes for three methods

Another variable that is used for evaluating the performance of the scheduling process is turnaround time. The following Fig. 9 represents the average turnaround time for the processes.

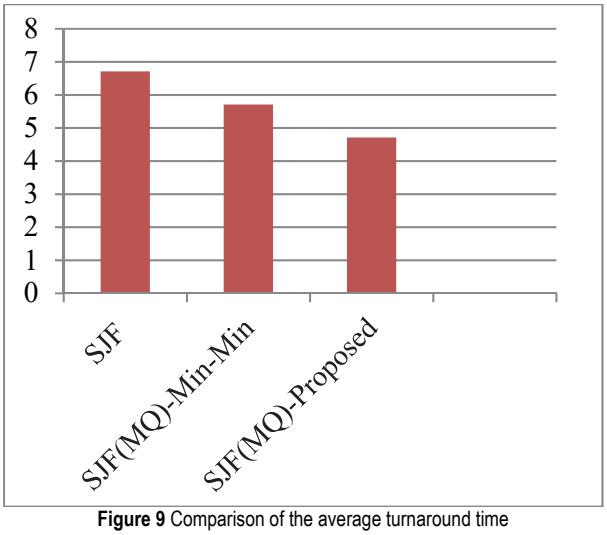

Figure 9 Comparison of the average turnaround time

The turnaround time of the individual processes is given in the following Fig. 10.

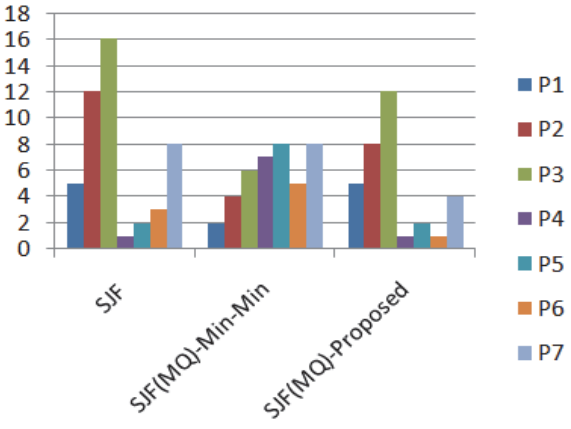

Figure 10 Turnaround time of the individual processes

Based on the results obtained from the implementation, it is evident that the proposed algorithm exhibits superior performance compared to the standard algorithm. Specifically, it outperforms the standard algorithm by an average of 16.71%, 14.45%, and 11.99% in terms of Makspan, resource utilization, and execution cost, respectively. Despite the higher time complexity associated with the proposed algorithm, it still manages to surpass the standard algorithm in the aspects of Makspan, execution cost, and resource utilization.

5 CONCLUSIONS

The scheduling approach under consideration amalgamates the shortest job first and Min-Min scheduling strategies while also addressing the issue of starvation through a fresh method for determining the optimal fit. The proposed model also exhibits better average waiting time and average turnaround time. Since average waiting time and turnaround time are key determining factors in the performance of the cloud data center, it is evident that the proposed approach will indeed enhance it. Implementation results demonstrate that the proposed algorithm outperforms the standard optimization approach in terms of makespan, resource utilization, and cost. Regrettably, this enhancement has come at the cost of increased time complexity. Future improvements should be developed with a keen awareness of the unique demands and obstacles within the healthcare sector. These considerations must encompass patient confidentiality, data protection, and the high-stakes nature of medical applications. Ongoing updates and the ability to adapt to emerging technologies and evolving healthcare regulations are imperative to ensure the algorithm remains effective in a dynamically changing environment. We intend to enhance the optimization algorithm by exploring alternative greedy algorithms and incorporating the consideration of task dependencies rather than assuming task independence. Our goal is to achieve heightened overall system performance.

6 REFERENCES

[1] Mian, G., Quansheng, G., & Wende, K. (2018). Optimal Scheduling of VMs in Queueing Cloud Computing Systems with a Heterogeneous Workload. *IEEE Access*, *6*, 15178-15191. https://doi.org/10.1109/ACCESS.2018.2801319

[2] Sahni, J. & Vidyarthi, D. (2018). A Cost-Effective Deadline-Constrained Dynamic Scheduling Algorithm for Scientific Workflows in a Cloud Environment. *IEEE Transactions on Cloud Computing*, *6*(1), 2-18. https://doi.org/10.1109/TCC.2015.2451649

[3] Arapakis, I., Park, S., & Pielot, M. (2021). Impact of Response Latency on User Behaviour in Web search. *Proceedings of the ACM SIGIR*, 103-112. https://doi.org/10.48550/arXiv.2101.09086

[4] Elsherbiny, S., Eldaydamony, E., Alrahmawy, M., & Reyad, A. E. (2018). An Enhanced Task Scheduling Algorithm on Cloud Computing Environment. *Egyptian Informatics Journal, International Journal of Grid and Distributed Computing*, *9*(7), 91-100. https://doi.org/10.14257/ijgdc.2016.9.7.10

[5] Divya, C. & Bijendra, K. (2019). Cloudy GSA for load Scheduling in cloud computing. *Applied Soft Computing*, *83*(C), 861-871. https://doi.org/10.1016/j.asoc.2019.105627

[6] Nabi, S., Ahmad, M., Ibrahim, M., & Hamam, H. (2022). AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing. *Sensors*, *22*(3), 920.

https://doi.org/10.3390/s22030920
[7] Rani, R. & Ritu, G. (2021). Energy efficient task scheduling using adaptive PSO for cloud computing. *International Journal of Reasoning-based Intelligent Systems*, *13*(2), 50-58. https://doi.org/10.1504/IJRIS.2021.114630
[8] Nabi, S. & Ahmed, M. (2021). PSO-RDAL: Particle swarm optimization-based resource- and deadline-aware dynamic load balancer for deadline constrained cloud tasks. *The Journal of Supercomputing*, *78*, 4624-4654.
[9] Mangalampalli , S., Swain, S. K., & Mangalampalli, V. K. (2021). Multi Objective Task Scheduling in Cloud Computing Using Cat Swarm Optimization Algorithm. *Arabian Journal for Science and Engineering*, *47*, 1821-1830. https://doi.org/10.1007/s13369-021-06076-7
[10] Malik, M. & Suman (2022). Lateral Wolf Based Particle Swarm Optimization (LW-PSO) for Load Balancing on Cloud Computing. *Wireless Personal Communications*, *125*(2), 1125-1144. https://doi.org/10.1007/s11277-022-09592-3
[11] Ankita & Sahana, S. K. (2021). Ba-PSO: A Balanced PSO to solve multi-objective grid scheduling problem. *Applied Intelligence*, *52*, 4015-4027.
[12] Agarwal, M. & Srivastava, G. M. S. (2021). Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing. *Journal of Ambient Intelligence and Humanized Computing*, *12*, 9855-9875.
[13] Shukri S. E., Al-Sayyed R., Hudaib, A., & Mirjalili, S. (2020). Enhanced multi-verse optimizer for task scheduling in cloud computing environments. *Expert Systems with Applications*, *168*, 114230. https://doi.org/10.1016/j.eswa.2020.114230
[14] Walia, N. K., Kaur, N., Alowaidi, M., Bhatia, K. S., Mishra, S., Sharma, N. K., Sharma, S. K., & Kaur, H. An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments. *IEEE Access*, *9*, 117325-117337. https://doi.org/10.1109/ACCESS.2021.3105727
[15] Hussain, M., Wei, L. F., Lakhan, A., Wali, S., Ali, S., & Hussain, A. (2021). Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. *Sustainable Computing: Informatics and Systems*, *30*, 100517. https://doi.org/10.1016/J.SUSCOM.2021.100517
[16] Velliangiri, S., Karthikeyan, P., Xavier, V. A., & Baswaraj, D. (2020). Hybrid electro search with genetic algorithm for task scheduling in cloud computing. *Ain Shams Engineering Journal*, *12*, 631-639. https://doi.org/10.1016/j.asej.2020.07.003
[17] Kanwal, S., Iqbal, Z., Al-Turjman, F., Irtaza, A., & Khan, M. A. (2021). Multiphase fault tolerance genetic algorithm for vm and task scheduling in datacenter. *Information Processing and Management*, *58*, 102676. https://doi.org/10.1016/j.ipm.2021.102676
[18] Shuaipeng, Y., Bailin, W., & Tike, L., (2023). An Iterated Greedy Algorithm for a Parallel Machine Scheduling Problem with Re-entrant and Group Processing Features. *Technical Gazette*, *30*(4), 1241-1252. https://doi.org/10.17559/TV-20230602000692
[19] Sanaj, M. S. & Joe Prathap, P. M. (2021). An efficient approach to the map-reduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment. *Mater Today Proceed*, *37*(2), 3199-3208. https://doi.org/10.1016/j.matpr.2020.09.064
[20] Pirozmand, P., Hosseinabadi, A. A. R., Farrokhzad, M., Sadeghilalimi, M., Mirkamali, S., & Slowik, A. (2021). Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing. *Neural Computing and Applications*, *33*, 13075-13088.
[21] Pirozmand, P., Javadpour, A., Nazarian, H., Pinto, P., Mirkamali, S., & Ja'Fari, F. (2022). GSAGA: A hybrid algorithm for task scheduling in cloud infrastructure. *The Journal of Supercomputing*, 78, 17423-17449. https://doi.org/10.1007/s11227-022-04539-8
[22] Mirmohseni, S. M., Tang, C., & Javadpour, A. (2022). FPSO-GA: A Fuzzy Metaheuristic Load Balancing Algorithm to Reduce Energy Consumption in Cloud Networks. *Wireless Personal Communications*, *127*, 2799-2821. https://doi.org/10.1007/s11277-022-09897-3
[23] Elsedimy, E. & Fahad, A. (2022). MOTS-ACO: An improved ant colony optimiser for multi-objective task scheduling optimisation problem in cloud data centres. *IET Net*, 11, 43-57. https://doi.org/10.1049/ntw2.12033
[24] Wei, Y., Pan, L., Shijun, L., Lei, W., & Xiangxu, M. (2018). DRL-Scheduling: an Intelligent QoS-Aware Job Scheduling Framework for Applications in Clouds. *IEEE access*, *6*, 55112 -55125. https://doi.org/10.1109/ACCESS.2018.2872674
[25] Huiyong, L., Shuhe, H., Xiaofeng, W., & Furong, W. (2023). A Novel Task of Loading and Computing Resource Scheduling Strategy in Internet of Vehicles Based on Dynamic Greedy Algorithm. *Technical Gazette*, *30*(3), 1298-1307. https://doi.org/10.17559/TV-20221207032927

Contact information:

Dr. Ganapriya K, Assistant Professor
(Corresponding author)
Department of ECE,
SSM Institute of Engineering and Technology,
Dindigul, India
E-mail: ganapriyag@gmail.com

Dr. Poobalan A, Assistant Professor
Department of Computer Science and Engineering,
University College of Engineering,
Dindigul, India
E-mail: apoobal@gmail.com

Dr. Gopinath S, Professor
Department of Department of ECE,
Karpagam Institute of Technology,
Coimbatore, India
E-mail: gopi.vasudev@gmail.com

Vedha Vinodha D, Assistant Professor
Department of Department of ECE,
JCT College of Engineering and Technology,
Coimbatore, India
E-mail: vedhavinodha.d@jct.ac.in