The Scientific Temper (2024) Vol. 15 (2): 2143-2151
E-ISSN: 2231-6396, ISSN: 0976-8653
Doi: 10.58414/SCIENTIFICTEMPER.2024.15.2.26
https://scientifictemper.com/

RESEARCH ARTICLE

Hazard regressive multipoint elitist spiral search optimization for resource efficient task scheduling in cloud computing

A. Jabeen[1*], A. R. M. Shanavas[2]

Abstract

Cloud computing, a revolutionary paradigm, connects computing resources and users via the internet, offering services like cloud storage and on-demand computing. In this context, efficient task scheduling is crucial, aiming to minimize costs and makespan. We introduce hazard regressive multipoint elitist spiral search optimization (HRMESSO), a novel technique for efficient task scheduling with reduced time consumption. User tasks are initially received, prioritized, and optimized. Cox proportional hazard regression is applied to establish relationships between task attributes (e.g., priority level, request arrival time, file size) and task prioritization. The great deluge elitist spiral search optimization identifies optimal virtual machines, considering factors like energy, memory, and CPU availability. The spiral search employs logarithmic spirals and Jenson Shannon divergence to find global optimal virtual machines. Finally, the task assigner schedules prioritized tasks onto the identified optimal virtual machines. Experimental evaluation is conducted with different metrics such as task scheduling efficiency, makespan, throughput and energy consumption. The quantitatively compared results exhibit the HRMESSO technique provides better scheduling efficiency, lesser makespan, throughput and energy consumption.

Keywords: Cloud computing, Resource optimization, Task scheduling, Cox proportional hazards regression, Great deluge elitist spiral search optimization, Jenson Shannon divergence.

Introduction

Cloud computing is a distributed computing model that enables users to access on-demand computing services, including storage, databases, and software, through the internet. The efficiency of task scheduling in the cloud addresses issues such as long scheduling times, high-cost consumption, and heavy virtual machine loads. Consequently, the task of scheduling tasks in a cloud computing environment remains a challenge, to access a quick and efficient solution. To tackle task scheduling in cloud computing, numerous methods have proposed cloud computing task scheduling models that leverage metaheuristic algorithms.

Multi objective trust aware task scheduling algorithm using whale optimization (MOTSWAO) was designed by Mangalampalli, S., Karri, G. R., & Kose, U., (2023). This algorithm aims to efficiently allocate tasks to appropriate virtual resources to reduce both makespan and energy consumption. However, the machine learning model was not implemented to predict upcoming workloads and improve the scheduling efficacy. Multi-objective algorithm integrates hybrid artificial bee colony and Q-learning (MOABCQ) was designed (Kruekaew, B., & Kimpan, W., 2022) to reduce the makespan, cost, and increase throughput and average resource utilization.The principal component gradient round-robin load balancing algorithm was designed by Mathanraj, E., & Reddy, R. N. (2024) to improve load balancing efficiency with maximum throughput and lesser makespan, migration cost, and response time. However, the desired higher level of efficiency was not achieved.

A metaheuristic framework was developed (Alsadie, D.,2021) to dynamically schedule the task to the virtual machine. However, an optimal strategy for selecting and placing virtual machines in the cloud computing

[1]Department of Computer Science, Cauvery College for Women (Autonomous), (Affiliated to Bharathidasan University) Thiruchirappalli, Tamil Nadu, India.

[2]Department of Computer Science, Jamal Mohamed College (Autonomous), (Affiliated to Bharathidasan University), Thiruchirappalli, Tamil Nadu, India.

***Corresponding Author:** A. Jabeen, Department of Computer Science, Cauvery College for Women (Autonomous), (Affiliated to Bharathidasan University) Thiruchirappalli, Tamil Nadu, India., E-Mail: jabeen.ca@cauverycollege.ac.in

How to cite this article: Jabeen, A., Shanavas, A. R. M. (2024). Hazard regressive multipoint elitist spiral search optimization for resource efficient task scheduling in cloud computing. The Scientific Temper, **15**(2):2143-2151.

Doi: 10.58414/SCIENTIFICTEMPER.2024.15.2.26

Source of support: Nil

Conflict of interest: None.

Received: 12/04/2024 **Accepted:** 12/05/2024 **Published** : 15/06/2024

environment was not developed. A parallel genetic algorithm with a MapReduce design was developed in (Peng, Z., *et al.,* 2022) to distribute tasks across various priority queues. However, the intended reduction in execution time was not achieved. In addition, the effective utilization of available resources was not maximized. A novel hybrid genetic algorithm (HGA) was introduced in (Hussain, A. A., & Al-Turjman, F., 2022) for efficient task scheduling with better resource utilization. However, it failed to increase the throughput with the computations to get more smoothed efficient task scheduling results.

A modified particle swarm optimization algorithm was designed in Chaudhary, S., *et al. (*2023) for excessively scheduling the tasks with minimum makespan. However, the throughput was not improved. A pair-based task scheduling was designed in Panda, S. K., *et al.* (2022) with the aim of minimizing layover time. But an, efficient scheduling was not implemented with a large number of tasks.

An enhanced sunflower optimization (ESFO) algorithm was designed in (Emami, H., 2022) to enhance task scheduling with minimum energy consumption and makespan. However, the machine learning algorithm was not applied to enhance task scheduling. Distributed grey wolf optimizer (DGWO) was developed in Abed-Alguni, *et al.* (2021) to distribute the dependent tasks to virtual machines with computation and data transmission costs. Though the DGWO reduces the makespan, energy consumption was not reduced. A fuzzy self-defense algorithm was designed in (Guo, X., 2021) for multi-objective task scheduling with resource utilization. However, an optimization algorithm was not employed to enhance the task scheduling performance.

Related Works

A multi-objective scheduling framework was introduced in (Reddy, P. V., & Reddy, K. G., 2023) using linear scaling-crow search optimization algorithm. However, the framework failed to extend with some advanced neural networks and perform the workflow scheduling across multiple clouds. A hybrid swarm optimization algorithm (Eldesokey, H. M., *et al.,* 2021) was designed to resolve task scheduling issues with the available resource. The primary objective was to minimize both execution time and computational costs. However, the work failed to extend with QoS parameters to perform task scheduling.

Task scheduling based queue algorithm was designed in (Lu, S., Gu, R., Jin, H., Wang, L., Li, X., & Li, J., 2021) with the aim of minimizing the data transmission delay and server load. But the energy aware task scheduling was major challenging issue. The whale optimization algorithm (WOA), introduced in (Chen, X., *et al.,* 2020) for optimizing cloud task scheduling with multi-objective functions. Its primary objective is to enhance the performance of a cloud system by efficiently allocating the computing resources. However, accuracy in task scheduling was not enhanced.

An artificial bee colony (ABC) algorithm was designed in (Babadi, M. S., Shiri, M. E., Goudarzi, M. R. M., & Javadi, H. H. S., 2022) for solving the task scheduling problem of cloud computing network with available processing resources. However, the performance of throughput was not improved. An integration of genetic algorithm (GA) and the gravitational emulation local search (GELS) algorithm were designed in (Praveen, S. P., Ghasempoor, H., Shahabi, N., & Izanloo, F., 2023) for task scheduling. But the performance of makespan was not reduced.

A priority assignment method was introduced in (Lipsa, S., *et al.,* 2023) for task scheduling problem using M/M/n queuing approach to assign priority to individual tasks. However it did not consider multi-objective task scheduling. An improved whale optimization algorithm (IWC) was designed in (Jia, L., *et al.,* 2021) for task scheduling and distribution model with lesser time and cost. But it failed to develop an efficient scheduling system for various task workloads.

Using artificial and convolutional neural network, a failure-aware task scheduling approach was introduced in (Alahmad, *et al.,* 2021). But the task scheduling efficiency was not minimized. A hybrid model was developed in (Gupta, P., *et al.,* 2023) based on meta-heuristic technique and neural network for optimal assignment of the tasks. However the designed model failed to attain higher throughput.

Proposed Methodology

Cloud computing is a highly demanding platform for managing user needs across the internet. The cloud computing system is constructed from a diverse range of distributed servers, data storage devices, and network infrastructures to provide highly manageable services. Task scheduling has emerged as a significant and challenging concern in cloud computing. Task scheduling entails the allocation of user tasks to servers that optimize performance and minimizes execution time. Resource utilization refers to the efficient utilization of available resources, which leads to improved system performance and cost savings. Therefore, task scheduling and resource utilization pose fundamental challenges across various domains to achieve optimal performance efficiency. With this motivation, the HRMESSO technique has been developed to address resource-optimized task scheduling in the cloud computing environment.

Figure 1 illustrates the architecture diagram of HRMESSO technique for resource-optimized task scheduling in cloud environment. The system model of task scheduling in cloud computing comprises four entities: user(U), cloud server (CS), task assigner (T_S) and virtual machines 'V_m'. First the cloud user 'U' submit the incoming tasks $T_1, T_2, T_3, \ldots . T_m$ to the cloud server (CS). In the cloud server, the task assigner (T_S) analyzes the incoming tasks and find the priority level using Cox proportional hazard regression. After finding the

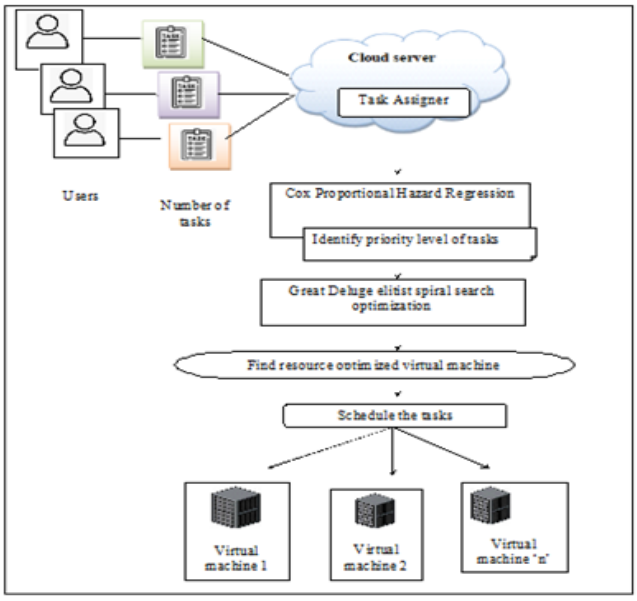

Figure 1: Architecture diagram of HRMESSO technique

priority, task scheduler determines the resource optimal virtual machines$V_1, V_2, V_3, \ldots V_m$ using great deluge elitist spiral search optimization. Then the assigner schedules the tasks to the optimal virtual machine.

Cox Proportional Hazard Regression

Priority-based task scheduling in cloud computing involves assigning priorities to different tasks based on their importance or urgency and then scheduling these tasks on available computing resources consequently. This approach ensures that high priority user requested tasks are completed in a timely manner than other this in turn enhances the overall system efficiency. Therefore, the proposed HRMESSO technique uses the Cox proportional hazard regression to identify the priority level of the arrived tasks.

Cox proportional hazard regression is a statistical model used for analyzing time-to-event data in the context of task scheduling. Cox regression is widely used for measuring the relationship between among dependent data, (i.e., priority level) and one or more independent data of cloud user requested task, (i.e., request arrival time, file size, predicted job completion time).

Let us consider the cloud user 'U' submit the incoming tasks $T_1, T_2, T_3, \ldots T_m$ in cloud server. The task scheduler first analyzes incoming tasks regarding request arrival time, file size, and predicted job completion time.

Request arrival time is a time at which a user's task or request arrives. This arrival time is often measured in seconds, minutes or hours.

File size is an important consideration in storage because it affects how much space a requested user consumes. It measured in terms of bytes, kilo bytes (KB), mega bytes (MB), giga bytes (GB). A larger file size is related to an increased likelihood of the task being assigned a high priority.

Predicted task completion time is an important independent variable affecting the task's priority. The tasks with shorter predicted completion times are taken higher priority. This completion time is typically based on various factors such as the task's complexity, historical data.

$$Pt_T = T_c - T_s \tag{1}$$

Where, Pt_T denotes a predicted task completion time, T_c a completion time of tasks, and a starting time of the particular user tasks. This time is often measured in seconds, minutes or hours.

In this way, the independent variables are estimated. Therefore, the Cox proportional hazard regression is applied to analyze the one or more independent data as given below,

$$R = h_0(t) * e^{[Q_1\beta_1 + Q_2\beta_2 + Q_3\beta_3]} \tag{2}$$

Where, R denotes a dependent variable, $h_0(t)$ denotes a base line hazard function with non negative, Q_1 denotes a request arrival time, Q_2 indicates a file size, Q_3 denotes a predicted task completion time, $\beta_1, \beta_2, \beta_3$ are the coefficients. From (2) R provides the outcomes values from 0 to 1 used for predicting outcomes such as high, medium and low priority based on independent variables.

$$Y = \begin{cases} 0 < R < 0.5; & Low \\ R = 0.5 \quad ; & Medium \\ 0.5 < R \leq 1; & High \end{cases} \tag{3}$$

Where, Y denotes a regression outcomes, 'R' indicates a dependent variable. If the dependent variable ranges from 0 to 0.5, a low priority is assigned to the task. When the dependent variable is exactly 0.5, a medium priority is assigned to the task. If the dependent variable falls between 0.5 and 1, a high priority is assigned to the task. This way, incoming user tasks are prioritized and stored in the queue. The algorithmic process of task prioritization is given below.

Algorithm 1 provided above outlines the step-by-step process of task scheduling in a cloud environment. When dealing with each incoming user-requested task, the task scheduler initially identifies independent variables, such as the task's arrival time, file size, and predicted completion time. Subsequently, the Cox proportional hazard regression is employed to analyze these independent variables. Based on the regression analysis results, the incoming tasks are categorized and prioritized as low, medium, and high priority. Finally, the prioritized tasks are stored in queue for subsequent task scheduling.

Great Deluge Multipoint Spiral Search Resource Optimization based Task Scheduling

Once the tasks get prioritized, the scheduling process is carried out in a resource optimal manner. Task scheduling in the context of cloud computing refers to the process

// **Algorithm 1:** Cox proportional hazard regression based task prioritization

Input: Number of cloud user tasks $T_1, T_2, T_3, \ldots . T_n$
Output: prioritizes the user tasks

Begin
Step 1: For each incoming task T_i
Step 2: Find the independent parameters request arrival time, File size and predicted task completion time
Step 3: Perform regression analysis 'R' using (2)
Step 4: If $(0 < R < 0.5)$ then
Step 5: The task is said to be a low priority
Step 6: else if $(R = 0.5)$ then
Step 7: The task is said to be a medium priority
Step 8: else if $(0.5 < R \leq 1)$ then
Step 7: The task is said to be a high priority
Step 8: End if
Step 9: Tasks are stored in the queue
Step 10: End for

of allocating tasks to execute a set of virtual machines efficiently in a distributed cloud infrastructure. This process also ensures optimal resource utilization to achieve desired performance, throughput, and resource utilization efficiency. The proposed HRMESSO technique uses the great deluge multipoint spiral search algorithm for resource efficient task scheduling in cloud.

Great deluge multipoint spiral search algorithm is a metaheuristic inspired by spiral phenomena in nature. The main objective of the algorithm is to generate logarithmic spirals share the diversification and intensification performance. The diversification behavior enables for global search (exploration) and the intensification behavior enables a search around a current solution (i.e. local search) (exploitation).

The spiral search algorithm is a multipoint search algorithm that uses multiple spiral models described as deterministic systems. Because search points track the logarithmic spiral route towards the common center, defined as the current best point. Then the global best solution is determined by updating the common center.

Great deluge algorithm starts with a broad search and progressively focuses on escaping local optima and finding better solutions. The global best solution is obtained by defining the number of neighbor's search points. The neighbor's search points are identified based on removing or adding the points from the current solution.

In this algorithm, multiple search points are related to the number of virtual machines in the cloud server. First, the multi = points (virtual machines) populations are initialized in search space.

$$V_m = V_1, V_2, V_3, \ldots . V_m \quad (4)$$

By applying a great deluge algorithm, neighbor's search points are initialized based on removing the points from the current solution.

$$V_k = V_1, V_2, V_3, \ldots . V_k \quad (5)$$

For every point in current and neighboring points, the fitness value is calculated based on multiple resources such as energy, memory, and CPU time.

Energy is a significant resource in task scheduling and resource allocation scenarios, particularly in the context of cloud computing. It involves optimizing the allocation and execution of tasks on computational resources.

First, the energy availability of a virtual machine is measured as follows

$$EG_{Avl} = EG_{Tot} - EG_{Com} \quad (6)$$

Where, EG_{Avl} represents an energy availability of the virtual machine, EG_{Tot} represents total energy of virtual machine, EG_{Com} be the consumed energy.

Memory is the significant resource in task scheduling. It is measured as the amount of storage space required to process the user requested tasks. Therefore, the memory availability is estimated to find out the storage availability of virtual machine. It is mathematically formulated as given below,

$$MEM_{Avl} = MEM_{Tot} - MEM_{Com} \quad (7)$$

Where, MEM_{Avl} indicates the memory availability of the virtual machine, MEM_{Tot} represents a total memory capacity of the virtual machine and MEM_{Com} denotes a consumed memory capacity of a virtual machine.

CPU time

This is the time the virtual machine spends executing user tasks.

$$CPU_{TM} = Time\ [E(T)] \quad (8)$$

From (8), CPU_{TM} which indicates the virtual machine's CPU time, $E(T)]$ represents a total time to execute the particular task.

The fitness of each solution and its neighboring solutions is computed by assessing the virtual machine's resources. This process helps identify both the current and global best solutions. The fitness measurement is performed as described below:

$$f = argma\,x\,[EG_{Avl}\ \&\&\ MEM_{Avl}] + arg\ min\ [CPU_{TM}] \quad (9)$$

Where,f represents a fitness, $arg\ max$ indicates an argument of a maximum function, $\arg min$ denotes an argument of minimum function.

The Elitist selection technique is applied to determine the current best solution as a center point. Then the elitist selection strategy is applied to determine the current best solution, (i.e. center point) based on fitness.

$$Z = \begin{cases} f(V_k) > f(V_m)\ ; & accept\ NP\ as\ CP \\ Otherwise\ ; & V_m\ as\ CP \end{cases} \quad (10)$$

Where, Z indicates elitist selection outcomes, $f(V_k)$ denotes a fitness of the neighboring virtual machine, $f(V_m)$ denotes a fitness of initial solution. As a result, the neighboring point (NP) with higher fitness than the initial solution is selected as the center point (CP). Otherwise, the higher fitness among the initial solution is selected as a center point (CP).

After selecting the center point, the position of the next search point gets updated as follows,

$$P_i(t+1) = P_i(t) + q_k R_a \frac{1}{2}|P_i(t) - CP_i(t)| \quad (11)$$

Where, $P_i(t+1)$ denotes a update next search point position, $P_i(t)$ indicates a current position of the search point, q_k denotes a step rate value from 0 to 1, R_a indicates a rotation matrix, i.e., identity matrix, $\frac{1}{2}|P_i(t) - CP_i(t)|$ Jensen Shannon divergence between the current position of search point '$P_i(t)$' and '$CP_i(t)$' indicates a position of the center point.

Followed by, the center point is updated as follows,

$$CP_i(t+1) = \begin{cases} CP_j(t+1); & if\ f\left(CP_j(t+1)\right) > f\left(CP_i(t)\right) \\ CP_i(t); & otherwise \end{cases} \quad (12)$$

Where, $CP_i(t+1)$ indicates an updated position of the center point, $CP_j(t+1)$ new position of the center point, $CP_i(t)$ indicates a current position of the center point. If the fitness of the updated position of the center point '$CP_j(t+1)$' is greater than current position of the center point '$CP_i(t)$', the position gets updated as '$CP_j(t+1)$. Otherwise, the previsions position is selected as final.

Figure 2 depicts the flowchart of the optimization technique aimed at selecting the optimal point (i.e. virtual machine) for scheduling incoming tasks. First, the defined search space generates a population of search points and their corresponding neighboring points. The fitness is evaluated for every point within this population, considering multiple objective functions. Subsequently, the center point is identified, and the positions of the search points are updated accordingly. This iterative process continues until the maximum iteration limit is attained. After reaching the maximum iteration, the task assigner schedules high-priority tasks onto the optimally selected virtual machine, thereby ensuring optimal efficiency. The algorithmic process of great deluge elitist spiral search optimization based task scheduling is shown in Figure 2.

Algorithm 2, as described above, outlines the various steps involved in task scheduling using the great deluge elitist spiral search optimization approach. The task assigner performs the scheduling process for each prioritized user task by identifying the resource-optimal virtual machine. The initial phase involves the random initialization of search points (i.e. virtual machines) in the search space for both the population of search points and neighboring search points. For each point within these populations, the proposed optimization technique assesses fitness based on multiple resources, including the virtual machine's energy, memory, and CPU time. Following fitness evaluation, the center point is selected using an elitist selection strategy with the highest fitness compared to others. Subsequently, the position of the next search point is updated. Followed by, the position of the center point is adjusted. This entire process iterates until the algorithm reaches its maximum specified number of iterations denoted by 't'. This iterative approach allows the spiral search algorithm to identify the optimal virtual machine. Finally, the task assigner allocates high-priority tasks to the selected optimal virtual machine to achieve improved resource utilization. This overall process enhances task scheduling efficiency and minimizes the

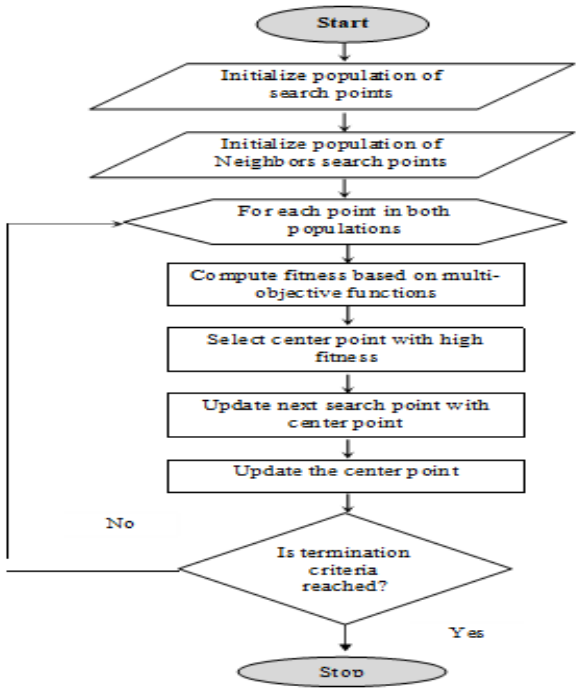

Figure 2: Flow chart of the great deluge elitist spiral search optimization

// **Algorithm 2:** Great deluge elitist spiral search optimization based task scheduling

Input: cloud server (CS), number of virtual machines $V_m = V_1, V_2, V_3, \ldots . V_m$, number of prioritized tasks $T_{ip} \in T_1, T_2, T_3, \ldots . T_m$,
Output: improve the task scheduling efficiency

Begin
Step 1: Initialize the population of the virtual machine $V_m = V_1, V_2, V_3, \ldots . V_m$
Step 2: Initialize the population of the neighbors' solution $V_k = V_1, V_2, V_3, \ldots . V_k$
Step 3: for each solution in both populations
Step 4: Estimate resource availability 'EG_{Avl}','MEM_{Avl}','CPU_{TM}'
Step 5 Calculate the fitness 'f'
Step 6: End for
Step 7: While (t < max_ iteration)
Step 8: Select the center point with high fitness using (10)
Step 9: Update the next search point based on the selected center point using (11)
Step 10: ***if*** $\left(f\left(CP_j(t+1)\right) > f\left(CP_i(t)\right)\right)$ then
Step 11: Update the center point as a $CP_j(t+1)$
Step 12: else
Step 13: Update the center point as a '$CP_i(t)$'
Step 14: t= t+1
Step 15: end while
Step 16: Obtain the optimal virtual machine
Step 17: Task manager schedules the tasks to the optimal virtual machine
End

makespan, contributing to more efficient task management and execution.

Experimental Analysis

Experimental evaluations of the proposed HRMESSO technique and existing methods namely MOTSWAO (Guo, X., 2021) and MOABCQ (Kruekaew, B., & Kimpan, W., 2022) are implemented in Java with the CloudSim network simulator. In order to conduct the experiment, personal cloud datasets is considered and take from http://cloudspaces.eu/results/datasets. The primary objective of this dataset is to facilitate the analysis of load and transfer workloads within a cloud environment. This dataset is comprised of 17 attributes (columns) and 66,245 instances. Among these attributes, two, namely time zone and capped, are excluded from consideration. The remaining 15 attributes are utilized for the purpose of resource-aware task scheduling across multiple virtual machines within the cloud server. These attributes include row id, account id, file size (task size), operation_time_start, operation_time_end, operation_id, operation type, bandwidth trace, node_ip, node_name, quoto_start, quoto_end, quoto_total (storage capacity), failed, and failure info.

Results and Discussion

This section presents the experimental evaluation of the HRMESSO technique alongside two existing methods, namely MOTSWAO (Guo, X., 2021) and MOABCQ (Kruekaew, B., & Kimpan, W., 2022) in terms of task scheduling efficiency, makespan, throughput and energy consumption. The performances of these parameters are analyzed using table and graphical representation.

Comparative Task Scheduling Efficiency Analysis

Task scheduling efficiency refers to the ratio of the number of successfully scheduled tasks to the total number of tasks. This efficiency is calculated using the following mathematical formula:

$$Eff_{TS} = \sum_{i=1}^{n} \frac{Tasks\ scheduled}{T_i} * 100 \quad (13)$$

From (13), Eff_{TS} denotes a task scheduling efficiency, $Tasks\ scheduled$ indiacates the number of tasks scheduled, 'n' represents the number of tasks 'T'. The efficiency is measured in percentage (%).

Table 1 and Figure 3 provide a graphical representation of task scheduling efficiency across distinct numbers of user tasks ranging from 1000 to 10000. The graph's horizontal axis represents the number of user tasks, while the vertical axis represents the task scheduling efficiency. Figure 3 compares the results of three different algorithms: HRMESSO, MOTSWAO (Guo, X., 2021), and MOABCQ (Kruekaew, B., & Kimpan, W., 2022). It is evident that the HRMESSO technique yields the highest task scheduling efficiency. This observation is validated through statistical evaluation. In an experiment involving 1000 user tasks, the

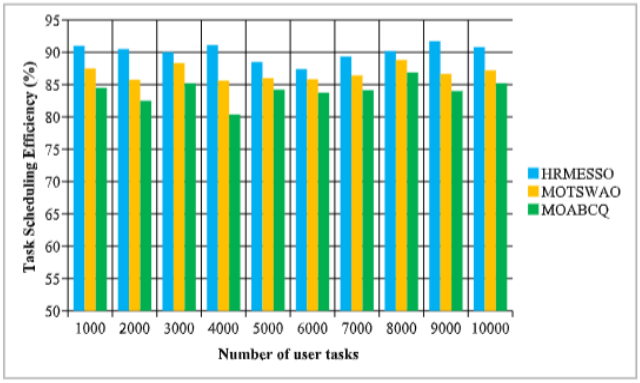

Figure 3: Comparison of algorithms in terms of task scheduling efficiency

Table 1: Comparative task scheduling efficiency analysis

Number of user tasks	*Task scheduling efficiency (%)*		
	HRMESSO	*MOTSWAO*	*MOABCQ*
1000	91	87.5	84.5
2000	90.5	85.75	82.5
3000	90	88.33	85.2
4000	91.12	85.62	80.37
5000	88.5	86	84.22
6000	87.4	85.83	83.75
7000	89.35	86.42	84.14
8000	90.17	88.81	86.87
9000	91.72	86.66	84
10000	90.8	87.2	85.2

HRMESSO technique achieved a task scheduling efficiency of 91%. In contrast, the efficiency of (Guo, X., 2021) and (Kruekaew, B., & Kimpan, W., 2022) was measured at 87.5% and 84%, respectively. Similar distinct efficiency results were obtained for each method. Comparing the performance outcomes, HRMESSO demonstrated a 7% improvement in task scheduling efficiency over (Guo, X., 2021) and a 3% improvement over (Kruekaew, B., & Kimpan, W., 2022). The application of the great deluge elitist spiral search optimization technique is introduced as the basis for these results. This optimization algorithm identifies the optimal virtual machine based on resource availability. By utilizing Jenson Shannon divergence, the optimization algorithm effectively identifies the global optimal virtual machine, enhancing the optimization process. Consequently, the task assigner within the cloud server schedules incoming tasks to this selected optimal virtual machine, improving efficiency.

Comparative Makespan Analysis

Makespan refers to the total duration to complete a set of tasks in a scheduling problem. It represents the overall time

required to execute a sequence of tasks. It is an important metric in various scheduling and optimization problems to minimize the makespan to achieve efficient resource utilization and task completion.

$$MKS = \sum_{i=1}^{n} T_i * tme\ (SST) \quad (14)$$

Where 'MKS' represents the makespan, 'n' number of tasks 'T_i' and $tme\ (SST)$ indicates a time for scheduling single task. It is measured in terms of milliseconds (ms).

Table 2 and Figure 4, presented above, display the performance results of the makespan achieved through three distinct methods: HRMESSO, MOTSWAO (Guo, X., 2021), and MOABCQ (Kruekaew, B., & Kimpan, W., 2022). To compute the makespan, a range of task collected from 1000 to 10000 in the dataset. The table reveals that the proposed HRMESSO technique delivers distinct results when compared to existing methods. In an experiment involving 1000 user tasks, the HRMESSO technique exhibited a makespan performance of 110ms , while (Guo, X., 2021) and (Kruekaew, B., & Kimpan, W., 2022) demonstrated makespans of 122 ms and 130ms, respectively. Similar variations in performance outcomes were observed across varying quantities of cloud user tasks. This validation outcome underscores that the HRMESSO technique minimizes an overall time consumption reduction of 15% when compared to (Guo, X., 2021), and 7% compared to (Kruekaew, B., & Kimpan, W., 2022). This improvement is realized through an effective task assigner to identify resource-efficient virtual machines. Initially, the task assigner employs Cox proportional hazard regression to distinguish user-requested tasks, relying on the analysis of dependent and independent data relating to cloud user requested tasks, including arrival time, file size, and task completion time. This analysis results in task prioritization and queuing. Subsequently, the task assigner identifies resource-efficient virtual machines and schedules tasks with minimal time consumption.

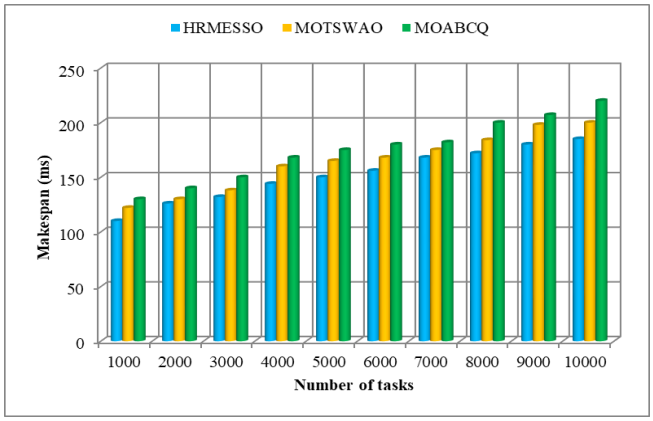

Figure 4: Comparison of algorithms in terms of makespan

Comparative Throughput Analysis

Throughput refers to the quantity of user-requested tasks completed within a specified unit of time in a cloud environment. Maximizing throughput helps in achieving higher task completion rates.

$$TPT = \left[\frac{Number\ of\ tasks\ completed}{t\ (sec)}\right] \quad (15)$$

Where TPT denotes a throughput, t denotes a time in second (sec), and it measured in terms of tasks per second (tasks/sec).

Table 3 and Figure 5 present a comparative performance analysis of throughput versus the number of user tasks gathered from datasets of 1000 and 10000 tasks. In Figure 5, the horizontal axis represents the number of tasks, while the vertical axis indicates throughput performance. The results demonstrate that the proposed HRMESSO technique outperforms the existing methods (Guo, X., 2021, Kruekaew, B., & Kimpan, W., 2022). Upon analyzing the outcomes presented above, distinct throughput patterns were observed for all three methods. HRMESSO achieved the highest throughput among these methods compared to the other existing approaches. For each method, ten separate comparisons were conducted to ensure robustness.

Table 2: Comparative makespan analysis

Number of user tasks	*Makespan (ms)*		
	HRMESSO	*MOTSWAO*	*MOABCQ*
1000	110	122	130
2000	126	130	140
3000	132	138	150
4000	144	160	168
5000	150	165	175
6000	156	168	180
7000	168	175	182
8000	172	184	200
9000	180	198	207
10000	185	200	220

Table 3: Comparative throughput analysis

Number of user tasks	*Throughput (tasks/sec)*		
	HRMESSO	*MOTSWAO*	*MOABCQ*
1000	215	185	178
2000	356	286	246
3000	455	375	325
4000	685	455	442
5000	725	685	596
6000	815	725	675
7000	905	875	725
8000	1022	974	865
9000	1128	1025	978
10000	1260	1125	1023

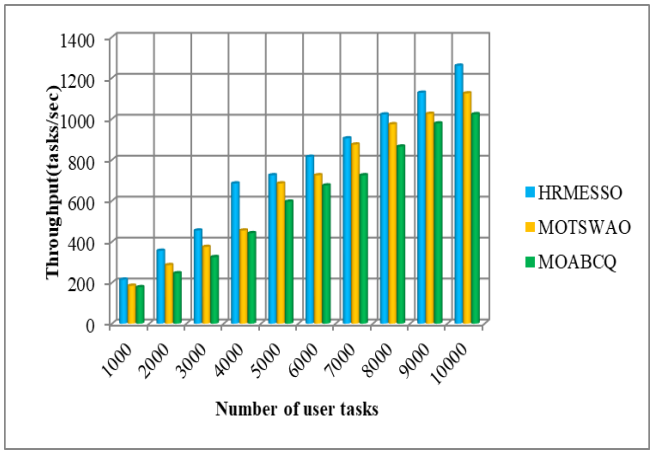

Figure 5: Comparison of algorithms in terms of throughput

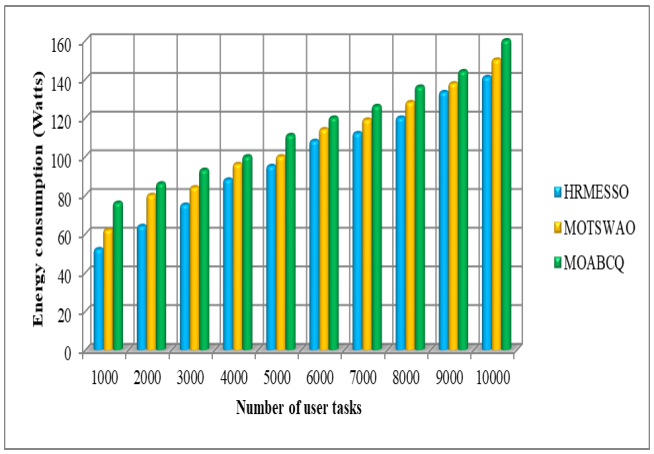

Figure 6: Comparison of algorithms in terms of energy consumption

Finally, the average of these ten comparisons reveals that the throughput performance using HRMESSO improved by 16% compared to (Guo, X., 2021) and 28% compared to (Kruekaew, B., & Kimpan, W., 2022). This achievement is occurred due to the application of the great deluge elitist spiral search optimization algorithm, which enables the identification of resource-efficient virtual machines. The proposed optimization algorithm efficiently selects virtual machines with sufficient energy availability, memory resources, and minimal CPU time utilization, as measured by the fitness metric. As a result, these chosen resource-efficient virtual machines demonstrate the capability to consistently execute numerous tasks within predefined time intervals.

Comparative Energy Consumption Analysis

Energy consumption is a significant concern in cloud computing for achieving the efficient task scheduling. It measured the amount of energy consumed by virtual machine to execute the number of tasks.

$$EC = \sum_{i=1}^{n} T_i * EG\ (ET) \quad (16)$$

Table 4: Comparative energy consumption analysis

Number of user tasks	*Energy consumption (Watts)*		
	HRMESSO	*MOTSWAO*	*MOABCQ*
1000	52	62	76
2000	64	80	86
3000	75	84	93
4000	88	96	100
5000	95	100	111
6000	108	114	120
7000	112	119	126
8000	120	128	136
9000	133.2	137.7	144
10000	141	150	160

Where 'EC represents the energy consumption, 'n' number of tasks 'T_i' and $EG\ (ET)$ indicates an energy consumed for executing the single task. Energy consumption is measured in terms of Watts.

Table 4 and Figure 6 present a performance analysis of energy consumption using HRMESSO, MOTSWAO (Guo, X., 2021), and MOABCQ (Kruekaew, B., & Kimpan, W., 2022). The evaluation involves measuring energy consumption across a range of cloud user tasks, varying from 1000 to 10000. The outcomes clearly demonstrate that the proposed HRMESSO technique outperforms the existing methods. In a specific experiment involving 1000 user tasks, the HRMESSO technique consumed 52 Watts energy and energy consumption using existing (Guo, X., 2021) and (Kruekaew, B., & Kimpan, W., 2022) methods were observed to be 62 Watts and 76 Watts, respectively. Each method's performance outcomes were observed based on the number of cloud user tasks. The validation results underline a notable reduction in energy consumption using the HRMESSO technique minimized by 7% compared to (Guo, X., 2021) and 15% compared to (Kruekaew, B., & Kimpan, W., 2022). The integration of the great deluge elitist spiral search optimization algorithm achieves this improvement. Each virtual machine's is assessed using multiple objective functions, allowing the selection of more energy-efficient virtual machines through the elitist selection strategy. As a result, these resource-efficient virtual machines execute multiple tasks more efficiently, consuming less energy.

Conclusion

A novel technique called HRMESSO aims to enhance the efficiency of task scheduling in the cloud. The HRMESSO technique utilizes cox proportional hazard regression to prioritize tasks based on factors such as request arrival time, file size and predicted completion time. Subsequently, the great deluge elitist spiral search optimization algorithm is employed to identify an optimal virtual machine with

improved resource availability for scheduling tasks with higher efficiency.

Experimental analysis is conducted to evaluate the performance of the HRMESSO technique and compare it with MOTSWAO and MOABCQ methods using various metrics, including task scheduling efficiency, makespan, throughput, and energy consumption. The implementation and the results demonstrate that the proposed HRMESSO technique achieved higher task scheduling efficiency, reduced makespan, and minimized energy consumption.

Acknowledgment

We sincerely acknowledge the Research convenor, Dr.A.R.Mohamed Shanavas, and Dr. D. I. George Amalarethinam, Principal of the institution, for providing the facility to complete this paper successfully.

References

Abed-Alguni, B. H., & Alawad, N. A. (2021). Distributed Grey Wolf Optimizer for scheduling of workflow applications in cloud environments. *Applied Soft Computing*, *102*, 107113. https://doi.org/10.1016/j.asoc.2021.107113

Alahmad, Y., Daradkeh, T., & Agarwal, A. (2021). Proactive failure-aware task scheduling framework for cloud computing. *IEEE Access, 9*, 106152-106168.DOI: 10.1109/ACCESS.2021.3101147

Alsadie, D. (2021). A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers. *IEEE Access, 9*, 74218-74233.DOI: 10.1109/ACCESS.2021.3077901

Babadi, M. S., Shiri, M. E., Goudarzi, M. R. M., & Javadi, H. H. S. (2022). Multi-objective tasks scheduling using bee colony algorithm in cloud computing. *International Journal of Electrical and Computer Engineering (IJECE), 12(5),* 5657-5666. DOI:10.11591/ijece.v12i5

Chaudhary, S., Sharma, V. K., Thakur, R. N., Rathi, A., Kumar, P., & Sharma, S. (2023). Modified Particle Swarm Optimization Based on Aging Leaders and Challengers Model for Task Scheduling in Cloud Computing. *Mathematical Problems in Engineering, 2023*. https://doi.org/10.1155/2023/3916735

Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y., & Murphy, J. (2020). A WOA-based optimization approach for task scheduling in cloud computing systems. *IEEE Systems journal, 14(3),* 3117-3128. DOI: 10.1109/JSYST.2019.2960088

Eldesokey, H. M., Abd El-atty, S. M., El-Shafai, W., Amoon, M., & Abd El-Samie, F. E. (2021). Hybrid swarm optimization algorithm based on task scheduling in a cloud environment. *International Journal of Communication Systems, 34(13),* e4694. https://doi.org/10.1002/dac.4694

Emami, H. (2022). Cloud task scheduling using enhanced sunflower optimization algorithm. *Ict Express, 8(1),* 97-100. https://doi.org/10.1016/j.icte.2021.08.001

Guo, X. (2021). Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm. *Alexandria Engineering Journal, 60(6),* 5603-5609.https://doi.org/10.1016/j.aej.2021.04.051

Gupta, P., Rawat, P. S., kumar Saini, D., Vidyarthi, A., & Alharbi, M. (2023). Neural network inspired differential evolution based task scheduling for cloud infrastructure. *Alexandria Engineering Journal, 73,* 217-230.https://doi.org/10.1016/j.aej.2023.04.032

Hussain, A. A., & Al-Turjman, F. (2022). Hybrid Genetic Algorithm for IOMT-Cloud Task Scheduling. *Wireless Communications and Mobile Computing, 2022*.https://doi.org/10.1155/2022/6604286

Jia, L., Li, K., & Shi, X. (2021). Cloud computing task scheduling model based on improved whale optimization algorithm. *Wireless Communications and Mobile Computing, 2021,* 1-13. https://doi.org/10.1155/2021/4888154

Kruekaew, B., & Kimpan, W. (2022). Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning. *IEEE Access, 10*, 17803-17818. DOI: 10.1109/ACCESS.2022.3149955

Lipsa, S., Dash, R. K., Ivković, N., & Cengiz, K. (2023). Task Scheduling in Cloud Computing: A Priority-Based Heuristic Approach. *IEEE Access, 11,* 27111-27126.DOI: 10.1109/ACCESS.2023.3255781

Lu, S., Gu, R., Jin, H., Wang, L., Li, X., & Li, J. (2021). QoS-aware task scheduling in cloud-edge environment. *IEEE Access, 9,* 56496-56505. DOI: 10.1109/ACCESS.2021.3072216

Mangalampalli, S., Karri, G. R., & Kose, U. (2023). Multi Objective Trust aware task scheduling algorithm in cloud computing using Whale Optimization. *Journal of King Saud University-Computer and Information Sciences, 35(2)*, 791-809.https://doi.org/10.1016/j.jksuci.2023.01.016

Mathanraj, E., & Reddy, R. N. (2024). Enhanced principal component gradient round-robin load balancing in cloud computing. *The Scientific Temper, 15(01)*, 1806–1815. DOI: https://doi.org/10.58414/SCIENTIFICTEMPER.2024.15.1.32

Panda, S. K., Nanda, S. S., & Bhoi, S. K. (2022). A pair-based task scheduling algorithm for cloud computing environment. *Journal of King Saud University-Computer and Information Sciences, 34(1),* 1434-1445.https://doi.org/10.1016/j.jksuci.2018.10.001

Peng, Z., Pirozmand, P., Motevalli, M., & Esmaeili, A. (2022). Genetic Algorithm-Based Task Scheduling in Cloud Computing Using MapReduce Framework. *Mathematical Problems in Engineering, 2022*. https://doi.org/10.1155/2022/4290382

Praveen, S. P., Ghasempoor, H., Shahabi, N., & Izanloo, F. (2023). A hybrid gravitational emulation local search-based algorithm for task scheduling in cloud computing. *Mathematical Problems in Engineering,* 2023.https://doi.org/10.1155/2023/6516482

Reddy, P. V., & Reddy, K. G. (2023). A Multi-objective based Scheduling Framework for Effective Resource Utilization in Cloud Computing. *IEEE Access*. DOI: 10.1109/ACCESS.2023.3266294