Journal of Statistics & Management Systems
ISSN 0972-0510 (Print), ISSN 2169-0014 (Online)
Vol. 26 (2023), No. 6, pp. 1377–1389
DOI : 10.47974/JSMS-1078

NDTAEP: Design of a novel deadline-aware task scheduling model using augmented ensemble pattern analysis

A. D. Gaikwad *
K. R. Singh †
Yeshwantrao Chavan College of Engineering
RTMNU Nagpur University Nagpur
Nagpur 440001
Maharashtra
India

S. D. Kamble §
Indira Gandhi Delhi Technical University for Women
Delhi 110006
India

Vikas Chouhan ‡
University of New Brunswick
Fredericton E3B5A3
New Brunswick
Canada

Abstract

A wide variety of scheduling models have been proposed by researchers over the years, and each of them has varying performance in terms of deadline hit ratio, scheduling effort, efficiency of task mapping, etc. However, these models are highly context-sensitive and cannot be scaled to heterogeneous task types due to their internal mapping characteristics. To improve task scalability, this work proposes a design of a novel deadline-aware task scheduling model that uses augmented ensemble pattern analysis for task clustering. The pattern analysis module uses a combination of K-means, hierarchical, and Fuzzy C Means (FCM) clustering to effectively segregate tasks depending on their completion and deadline parameters. These tasks are given to a modified deadline-aware League Championship Algorithm (LCA) optimizer, which assists in mapping the clustered tasks with worker

*E-mail: amolgaikwad.ag@gmail.com (Corresponding Author)
† E-mail: singhkavita19@yahoo.co.in
§ E-mail: Shaileshdkamble@igdtuw.ac.in
‡ E-mail: vchouhan@cs.iitr.ac.in

threads. The modified LCA model uses a combination of task priority, task deadline, and worker capacity for scheduling. The model maps tasks that require higher execution effort with moderately performing worker nodes, while tasks with nearer deadlines are allotted to higher-performance workers. Due to the use of an ensemble augmented pattern analyzer with a modified LCA optimizer, the proposed model can improve execution speed by 8%, deadline hit ratio by 1.5%, and scheduling efficiency by 6.5% when compared with various state-of-the-art scheduling approaches. The proposed model was evaluated and showcased a deadline hit ratio of 99.95%, computational efficiency of 96.26%, and average task-scheduling delay of less than 0.1 ms, which makes it highly useful for a wide variety of task scheduling application scenarios.

Subject Classification: *30D40, 62H30, 13F60.*

Keywords: *Cloud computing, Load balancing, Task scheduling, League championship algorithm, Pattern analysis.*

1. Introduction

Optimization of task scheduling models is a multidomain task involving effective fitness function design, rule optimization, iteration-based incremental analysis, etc. Back Propagation Neural Network (BPNN) based scheduling model analyzes task delays, cost of execution, temporal scheduling parameters, current task set, and other influencing factors [1]. These factors are given to a BPNN model, which assists in scheduling the tasks into public and private cloud-based execution queues. The results of this scheduling are fed back into the system, and depending upon the efficiency evaluated by this feedback, BPNN is retrained for better scheduling performance. Researchers have proposed a wide variety of such scheduling models over the years [2], and a survey of these models is discussed in the next section. Based on this survey, readers would be able to identify various nuances, advantages, limitations, and future research scopes for the reviewed models. This survey observed that task-scheduling models are highly context-aware and have limited applicability for heterogeneous types of tasks. Thus, Section 3 proposes a novel deadline-aware task scheduling model using augmented ensemble pattern analysis that aims to improve existing scheduling models' efficiency. This efficiency was evaluated in terms of scheduling delay, efficiency of scheduling, and deadline hit ratio; and was compared with various state-of-the-art approaches in Section 4, which assists in evaluating the run-time performance of the proposed model. Finally, this work concludes with some interesting observations about the proposed model and recommends various methods to improve its performance.

2. Literature Review

A wide variety of task scheduling models have been proposed by researchers over the years. Most of these models are context-sensitive and depend upon the type of tasks being scheduled[1]. For instance, the work in [3, 4] proposes makespan based on prediction and budget-constrained workflows-based models, which assist in scheduling tasks that have domain-specific constraints. These models have limited efficiency for general purpose task sets. To improve this efficiency machine learning, work in [5, 6] proposes Deep Neural Network with edge computing and resource-constrained profit optimization method on edge computing, which assist in high-speed scheduling computations. The efficiency of these computations can be improved via the work in [7], wherein Particle Swarm Optimization (PSO) is combined with idle time-slot aware rules. These rules assist in reducing redundancies during scheduling, while PSO assists in the identification of the best machine sets for the given set of tasks. Applications of these approaches can be observed from [8, 9, 10], wherein researchers have implemented minimized task completion time and execution cost models for smart grids, a self-adaptive model with learning automata, and metaheuristics for dynamic Virtual Machine (VM) allocation for cloud data centers. These applications assist in deploying context-specific scheduling engines, which improves the effectiveness of scheduling under constrained environments. Similar models are proposed in [11,12,13], wherein multiple organization task scheduling, hybrid estimation of distribution algorithm with Genetic Algorithm (EDA GA), and bandwidth-aware inter-cloud links for scheduling in hybrid clouds are discussed. These deployments are highly application specific but can be applied for general purpose scheduling with context-sensitive reconfigurations. These application-specific models are extended via the use of proactive failure-aware task scheduling [14], a combination of ant colony optimization with laxity (LACO) [15], multiple objective-based task scheduling [16], and Whale Optimization Algorithm (WOA) [17] models, which aim at applying metaheuristics for optimizing task-to-VM mapping efficiency for real-time environments. Similar models are proposed in [18, 19], wherein researchers have proposed Multiple Objective Grey Wolf Optimizer (MO GWO), multiple queue scheduling, and hierarchical with hybrid online task scheduling framework, which aims at optimizing multiple cloud-based task parameters, including the capacity of VM, task execution delay, deadline, etc. These models are generic and must be tuned as per the application under test for better scheduling results. Caching

is one of the techniques to retrieve the data faster which to improve the performance of the cloud [20].

3. Design of the proposed novel deadline-aware task scheduling model using augmented ensemble pattern analysis

The model initially clusters incoming tasks via a pattern analysis engine. This engine is a combination of K-means, fuzzy C Means, and hierarchical clustering models. It uses task length and the deadline for clustering, which assists in segregating tasks w.r.t. their completion parameters. The iterative LCA model intelligently selects an optimum learning rate, which can select worker-to-task mapping solutions that reduce the number of execution cycles. The selected learning rate is given to the final LCA model, which schedules pattern-analyzed tasks onto worker nodes. After the LCA model executes the tasks, an iterative performance-checking engine is activated. This engine evaluates the efficiency of task mapping and retrains the model for better worker configurations. The description of these modules is discussed in different sub-sections of this paper, which assists in implementing them in part(s) or as a whole, depending upon application requirements.

3.1 *Design of the pattern analysis module*

Incoming tasks from the dataset are initially processed via a pattern analysis module. This module evaluates task deadlines and task execution delays to cluster them via K-means, Fuzzy C Means (FCM), and hierarchical clustering methods. The results of these methods are combined via an ensemble layer. This layer outputs task patterns, which are further processed using a modified LCA model with learning rate optimization. All three clustering models utilize a Task Completion Metric (TCM), which is evaluated via Equation (1) as follows,

$$TCM_i = \frac{T_{delay_i}}{Max\left(\bigcup_{j=1}^{N_t} T_{Delay_j}\right)} + \frac{T_{deadline\,i}}{Max\left(\bigcup_{j=1}^{N_t} T_{Deadline_j}\right)} \quad ...(1)$$

Where T_{delay} $\&T_{deadline}$ represent delay and deadline needed for the completion of the task. These values are computed for each task and given to all clustering models. The K-means model works using the following process,

- Choose the number of clusters (k) and obtain the data points

- Randomly place the centroids. $c_1, c_2, \ldots c_k$
- Repeat the following process until convergence is achieved or until the end of a fixed number of iterations
- For each data point x_i, find the nearest centroid $(c_1, c_2, \ldots c_k)$
- Assign that cluster to the point cluster
- For each cluster assigned,

$$New_{centroid} = \frac{\sum_{i=1}^{T_e} x_i}{T_e} \quad \ldots(2)$$

Where T_e represents the total number of elements in that cluster. Similarly, FCM works via the following process,

- Choose the number of clusters (k) and obtain the data points
- Randomly place the centroids $c_1, c_2, \ldots c_k$
- For every cycle iteration
- Update membership function via Equation (3),

$$M_i = \frac{1}{\sum_{j=1}^{C} \left(\frac{[x_i - c_i]}{[x_i - c_j]} \right)^{\frac{2}{C}}} \quad \ldots(3)$$

- Calculate new centroids via Equation (4),

$$c_i = \frac{\sum_{j=1}^{C} M_i * x_j}{\sum_{j=1}^{C} M_j} \quad \ldots(4)$$

- Evaluate new objective function via Equation (5),

$$F = \Sigma_{i=1}^{C} \Sigma_{j=1}^{N} M_i \, |x_j - c_i|^2 \quad \ldots(5)$$

- Break this loop if the difference between the previous and current objective function is lower than the error threshold.

For hierarchical clustering, the following process is used,

- Select the (k) cluster and gather the data points.
- Divide the complete set of points into two clusters using the K-means method, and output the cluster with a lower number of elements.

- Select the cluster with the higher number of elements, and again apply K-means with two clusters, and repeat the previous step.
- Do this process $k - 1$ times, and output both clusters at the last split.

Based on these processes, tasks are divided into k clusters. All these models utilize an automatic cluster number evaluation engine, which assists in identifying the number of distinguishable clusters by observing the input dataset. This engine works using the following process,

- For a given clustering algorithm, set number of clusters C=2
- Perform clustering with these clusters, and evaluate inter-cluster distance via Equation (6),

$$Inter_d(i,j) = \sqrt{\sum_{a=1}^{N_p}(C_{a,i} - C_{a,j})^2} \quad ...(6)$$

Where $C_{a,i}$ represents the value of centroid for the i^{th} cluster, and N_p is the total number of features (or dimensions) for that centroid.

- Similarly, evaluate intra-cluster distance via Equation (7) as follows,

$$Intra_d(i) = \sqrt{\sum_{j=1}^{N_p}\sum_{a=1}^{N_p}(f_j(i) - f_a(i))^2} \quad ...(7)$$

Where f_i represents i^{th} feature vector for the current cluster.

- These values are evaluated by incrementing C in the range $C = 2 \ldots 10$ and finding the value of C, where maximum value of cluster difference (CD) is found.
- Equation (8) is used to calculate the value of CD as shown below,

$$CD = \frac{\sum_{i=1}^{C}\sum_{j=1}^{C} Inter_d(i,j)}{C * \sum_{i=1}^{C} Intra_d(i)} \quad ...(8)$$

Using this process, the number of clusters is estimated, and the input dataset is divided into multiple parts. Let the number of clusters selected by K-means be represented by N_k. After evaluation, it was observed that K-means had better clustering performance when compared with FCM and HC, thus, FCM and HC methods utilize value of N_k to perform clustering for segregating input tasks. Once the numbers of cluster are obtained, then the following process is evaluated to finalize the clusters,

- For each element i, if $kMeans_i = FCM_i = HC_i$, then the element is kept in the current cluster

- If $kMeans_i = FCM_i$, or $kMeans_i = HC_i$, then the element is shifted to cluster directed by K-means
- If $FCM_i = HC_i$, then the element is shifted to cluster directed by FCM
- Else, the element is shifted to the cluster directed by K-means

Based on this division, clusters are formed, and the results of these clusters are given to the modified LCA model for learning rate-optimized scheduling, which is discussed in the next section of this paper.

3.2 *Design of the modified LCA model for learning rate optimization*

Once the cluster formation process is performed by the pattern analysis, then the modified LCA model is applied, which assists in the pattern-based mapping of tasks with relevant worker threads. The modified model initially identifies the optimum learning rate for LCA, then uses this rate for scheduling the tasks. The internal method used by LCA can be described using the following process,

- Initialize the following LCA parameters,
  - Number of leagues (N_l)
  - Number of teams per match (N_t)
  - Number of seasons (N_S)
  - Initial learning rate ($I\,L_r$)
- During the initial iteration, perform the following tasks,
  - Designate all leagues as "to be reviewed" at the start.
  - Every season from one to N_S
  - Every League 1 to N_l
  - Proceed to the next league if the league is designated as "not to be reviewed."

Otherwise, create a new league by using the steps below:

- Generate a random number VMs, & sort VMs according to their capacity.

The capacity of a VM is evaluated via Equation (9),

$$C(VM) = MIPS(VM) * PE(VM) + \frac{BW(VM)}{MAX(BW)} + \frac{RAM(VM)}{MAX(RAM)} \quad ...(9)$$

Where *C, MIPS, PE, BW,* and *RAM* represent capacity, millions of instructions per second, number of processing elements, bandwidth, and memory available with the VM, respectively, which can be used for scheduling tasks.

- Sort VMs from highest to lowest capacity, and assign tasks from lowest cluster to highest cluster to these VMs
- Once all tasks are assigned, then evaluate the fitness of the league via Equation (10) as follows,

$$f = \frac{1}{N_i} \sum_{i=1}^{N_t} \frac{TCM_i}{C(VM)_i} \quad ...(10)$$

Where N_t represents the number of tasks, and *TCM* represents the task completion metric and is evaluated for each task.

- These fitness levels are assessed for each league, and Equation (11) is used to assess a fitness threshold as follows,

$$f_{th} = \sum_{i=1}^{N_l} \frac{f_i}{N_l} * IL_r \quad ...(11)$$

All leagues with fitness lower than f_{th} have the designation "not to be examined," while others have the designation "to be evaluated. "This procedure is used throughout all the seasons, and the league with the lowest fitness is chosen for iterative evaluation using Equation (12),

$$I_f = \frac{N_c}{TD} \quad ...(12)$$

Where N_C represents the number of cycles needed to execute this task, I_f represents iteration fitness, while *TD* represents task diversity, and is evaluated via Equation (13),

$$TD = \frac{\sum_{i=1}^{N_t} TCM_i - \sum_{j=1}^{N_t} \frac{TCM_j}{N_t}}{N_t} \quad ...(13)$$

Generate a new value of learning rate via Equation (14),

$$L_r = RAND(0.1,1) \quad ...(14)$$

- Repeat the entire LCA process with this value of L_R, and evaluate the new value of I_f
- Accept this value of L_r if,

$$New(I_f) < Old(I_f) \quad ...(15)$$

- Repeat this process for multiple iterations to obtain an optimum value of L_r

The process stops when the difference between the current and new iteration fitness is lower than a pre-set error threshold value. Based on this value of L_r LCA is executed, and mapping between VMs and their respective tasks is estimated. This mapping is continuously checked for correctness, and LCA is reiterated as per the process defined in the next section, where iteration performance-checking is performed.

3.3 *Design of iterative performance checker module*

Once tasks are mapped with their respective worker threads (or virtual machines), an iterative performance checker module is activated. This module evaluates the performance of the mapping for new tasks and either continues with the mapping or reiterates LCA for new mappings. During this phase, value of I_f is tracked for each batch of tasks, and its trend is evaluated. After every k^{th} batch, Equation (16) evaluates the iteration fitness threshold as follows,

$$Th_I = \sum_{i=1}^{B} I_{f_i} * \frac{L_r}{k} \quad ...(16)$$

After this evaluation, number of I_f values lower than Th_I are evaluated, and the LCA model is retrained if the following condition is fulfilled,

$$\frac{N_{I_f}}{k} > L_r \quad ...(17)$$

Where N_{I_f} represents the total number of times where I_f is lower than Th_I, which indicates that task diversity is low, or a higher number of computational cycles are needed to evaluate tasks. These checks determine whether to employ LCA in the chosen form or to retrain it for improved performance. In the following section of this work, performance is assessed in terms of execution speed, deadline hit ratio, and scheduling efficiency & contrasted with several cutting-edge scheduling systems.

4. Result evaluation and comparative analysis

The proposed paradigm enhances performance through continual learning improvement for better worker-to-task mapping. This performance is evaluated in terms of execution delay (D), deadline hit ratio (DHR), and scheduling efficiency (E), and compared with the models EDA GA [12], LACO [15], and MO GWO [18]. To perform this performance evaluation, parallel workloads archives were used. These include NASA logs, Los Alamos National Lab (LANL) CM-5 logs, SHARCNET logs, CIEMAT Euler logs, and KIT For HLR II logs. All these logs are available at https://www.cs.huji.ac.il/labs/parallel/workload and can be used with open-source licenses. The system was evaluated using the CloudSim simulator, which assists in Virtual Machine (VM) formation, along with task scheduling and performance evaluation of different models.

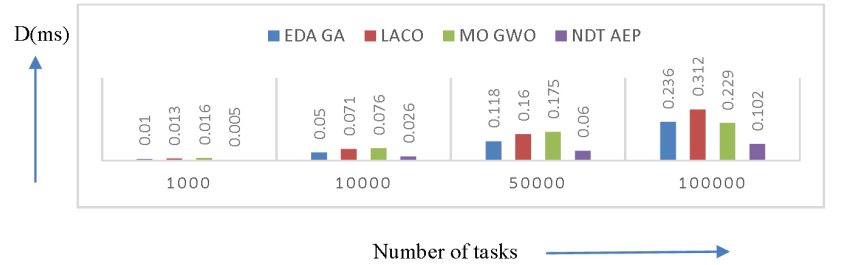

Figure 1

Execution delay for multiple tasks

This study and Figure 1 shows that the suggested model has an average task scheduling latency that is about 39% quicker than EDA GA [12], 46% faster than LACO [15], and 35% faster than MO GWO [18], making it suitable for high-speed scheduling applications.

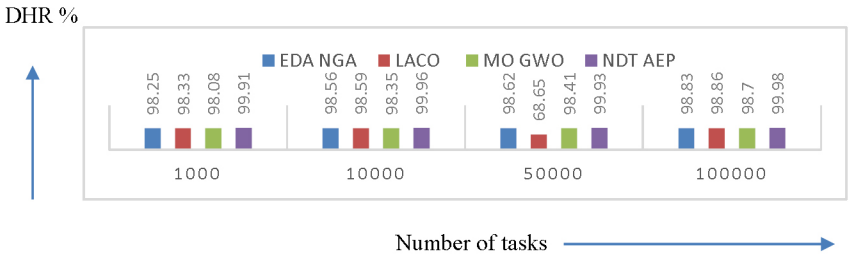

Figure 2

Deadline hit rate in % for different models

This assessment and Figure 2 shows that the suggested model is 1.2% more efficient than EDA GA [12], 1.15 percent more efficient than LACO [15], and 1.5% more efficient than MO GWO [18] in terms of DHR, making it an excellent choice for high-efficiency scheduling applications.

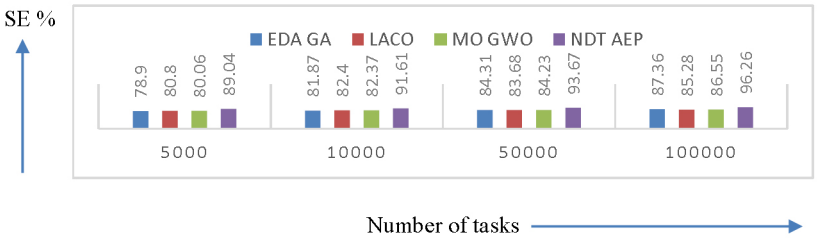

Figure 3

Scheduling efficiency for different models

This analysis and Figure 3 shows that the suggested model's scheduling efficiency is 9.2% better than that of EDA GA [12], 10.8% better than that of LACO [15], and 9.5% better than that of MO GWO [18], making it extremely beneficial for real-time scheduling applications. The usage of diverse tasks and the quantity of compute cycles during the assessment of iteration fitness are what led to this increase. Thus, the proposed model is capable of low delay, high DHR, and better efficiency scheduling application scenarios.

5. Conclusion and future scope

The suggested NDTAEP model combines pattern analysis with iterative modified LCA for enhanced scheduling performance. In order to iteratively improve scheduling performance, the suggested approach additionally employs a performance verification module. Because of this, the suggested model's average task scheduling latency is 39% faster than that of EDA GA [12], 46% faster than that of LACO [15], and 35% faster than that of MO GWO [18], making it effective for a wide range of scheduling applications. The proposed model is also 1.2% better in terms of DHR than EDA GA [12], 1.15 percent better than LACO [15], and 1.5% better than MO GWO [18], while it is 9.2% better in terms of scheduling efficiency than EDA GA [12], 10.8 percent better than LACO [15], and 9.5% better than MO GWO [18], making it extremely useful for real-time scheduling applications. Due to the use of deadline, task delay, task diversity, and VM capacity during scheduling, these benefits are made possible, making the proposed model extremely beneficial for a wide range of scheduling

scenarios. Furthermore, the suggested model can be applied to a variety of task kinds because it is independent of job type. Future studies could incorporate deep learning techniques like convolutional neural networks (CNNs), recurrent neural networks (RNNs), and others to further enhance the performance of the suggested model. The performance of the model can also be increased by incorporating different temporal pattern analysis engines. These engines can schedule activities based on the VMs' historical performance and scheduling abilities.

References

[1] Y. Xiong, S. Huang, M. Wu, J. She and K. Jiang, "A Johnson's-Rule-Based Genetic Algorithm for Two-Stage-Task Scheduling Problem in Data-Centers of Cloud Computing," in IEEE Transactions on Cloud Computing, vol. 7, no. 3, pp. 597-610 (2019).

[2] Vivek Gupta, Harpreet Singh Gill, Prabhdeep Singh & Rajbir Kaur, "An energy efficient fog-cloud based architecture for healthcare," in *Journal of Statistics and Management Systems*, vol. 21:4, pp. 529-537 (2018), DOI: 10.1080/09720510.2018.1466961.

[3] B. A. Al-Maytami, P. Fan, A. Hussain, T. Baker, and P. Liatsis, " A review of virtual machine (VM) resource scheduling algorithms in cloud computing environment," in *Journal of Statistics and Management Systems*, vol. 20:4, , pp. 703-711, DOI: 10.1080/09720510.2017.1395190

[4] F. Yao, C. Pu and Z. Zhang, "Task Duplication-Based Scheduling Algorithm for Budget-Constrained Workflows in Cloud Computing," in IEEE Access, vol. 9, pp. 37262-37272 (2021).

[5] Z. Chen, J. Hu, X. Chen, J. Hu, X. Zheng, and G. Min, "Computation Offloading and Task Scheduling for DNN-Based Applications in Cloud-Edge Computing," in IEEE Access, vol. 8, pp. 115537-115547, (2020).

[6] Kusum Tharani, Neeraj Kumar, Vishal Srivastava, Sakshi Mishra & M. Pratyush Jayachandran , Machine learning models for renewable energy forecasting,," in *Journal of Statistics and Management Systems*, vol. 23:1, pp. 171-180, (2020). DOI: 10.1080/09720510.2020.1721636.

[7] Y. Wang and X. Zuo, "An Effective Cloud Workflow Scheduling Approach Combining PSO and Idle Time Slot-Aware Rules," in *IEEE/CAA Journal of Automatic Sinica*, vol. 8, no. 5, pp. 1079-1094 (2021).

[8] H. Zhang, J. Shi, B. Deng, G. Jia, G. Han, and L. Shu, "MCTE: Minimizes Task Completion Time and Execution Cost to Optimize Scheduling Performance for Smart Grid Cloud," in IEEE Access, vol. 7, pp. 134793-134803 (2019).

[9] L. Zhu, K. Huang, Y. Hu, and X. Tai, "A Self-Adapting Task Scheduling Algorithm for Container Cloud Using Learning Automata," in IEEE Access, vol. 9, pp. 81236-81252 (2021).

[10] D. Alsadie, "A Metaheuristic Framework for Dynamic Virtual Machine Allocation with Optimized Task Scheduling in Cloud Data Centers," in IEEE Access, vol. 9, pp. 74218-74233 (2021).

[11] K. Dubey, M. Y. Shams, S. C. Sharma, A. Alarifi, M. Amoon and A. A. Nasr, "A Management System for Servicing Multi-Organizations on Community Cloud Model in Secure Cloud Environment," in IEEE Access, vol. 7, pp. 159535-159546 (2019).

[12] S. Pang, W. Li, H. He, Z. Shan, and X. Wang, "An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing," in IEEE Access, vol. 7, pp. 146379-146389 (2019).

[13] T. A. L. Genez, L. F. Bittencourt, N. L. S. d. Fonseca and E. R. M. Madeira, "Estimation of the Available Bandwidth in Inter-Cloud Links for Task Scheduling in Hybrid Clouds," in IEEE Transactions on Cloud Computing, vol. 7, no. 1, pp. 62-74 (2019).

[14] Y. Alahmad, T. Daradkeh and A. Agarwal, "Proactive Failure-Aware Task Scheduling Framework for Cloud Computing," in IEEE Access, vol. 9, pp. 106152-106168 (2021).

[15] J. Xu, Z. Hao, R. Zhang, and X. Sun, "A Method Based on the Combination of Laxity and Ant Colony System for Cloud-Fog Task Scheduling," in IEEE Access, vol. 7, pp. 116218-116226 (2019).

[16] S. Geng, D. Wu, P. Wang, and X. Cai, "Many-Objective Cloud Task Scheduling," in IEEE Access, vol. 8, pp. 79079-79088 (2020).

[17] X. Chen et al., "A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems," in *IEEE Systems Journal*, vol. 14, no. 3, pp. 3117-3128, (2020).

[18] D. Alsadie, "TSMGWO: Optimizing Task Schedule Using Multi-Objectives Grey Wolf Optimizer for Cloud Data Centers," in IEEE Access, vol. 9, pp. 37707-37725 (2021).

[19] H. Yuan, J. Bi and M. Zhou, "Multiqueue Scheduling of Heterogeneous Tasks With Bounded Response Time in Hybrid Green IaaS Clouds," in IEEE Transactions on Industrial Informatics, vol. 15, no. 10, pp. 5404-5412 (2019).

[20] Ruchi Nanda, Amita Sharma, Pooja Choraria, Astha Pareek, Neha Tiwari & Anubha Jain. Statistical analysis of query processing time in cache-based cloud database systems, *Journal of Statistics and Management Systems,* 25:7, 1673-1683 (2022), DOI: 10.1080/09720510.2022.2130576.

Received February, 2023