Achieving better Resource Utilization by Implementing a High-Performance Intelligent Framework in a Distributed Environment

Srinivasan Narayanasamy
Department of Computer Science and Engineering,
Rajalakshmi Engineering College, India
professorsrini@gmail.com

MohanKumar Palanichamy
Department of Computer Science and Engineering,
Hindustan Institute of Technology and Science, India
mohankumarmohan@gmail.com

Selvam Lakshmanan
Department of Computer Science and Engineering,
Karpagam Academy of Higher Education, India
umaselvam_35@yahoo.com

ArokiaRenjith Jerald
Department of Computer Science and Engineering,
Jeppiaar Engineering College, India
arokiarenjith@gmail.com

Abstract: *Multi-distributed high-performance computers from many companies are aggregated into a single computing platform to provide handlers with uniform contact besides convention outlines. Job arrangement strategies in High-Performance Computing (HPC) environments are lacking in flexibility, so an enhanced computational intelligence automated system in the task ready queue, refinement of the principal planner aimed at every job, and increased arrangement of the job setting up plan are proposed in this paper, which introduces an improved task scheduling model. The swarm intelligence method is used in core task scheduling to reduce the average scheduling time for completing tasks by assigning jobs to each node in the most efficient manner possible. The suggested scheduling technique outperforms the standard work scheduling approach in simulations. Task waiting times can be reduced, system throughput increased, task response times improved, and system resources better utilized by using a job setting up method created on group Acumen systems.*

Keywords: *Computational intelligence, scheduling, high-performance computing, resource utilization.*

Received February 8, 2022; accepted May 8, 2023
https://doi.org/10.34028/iajit/21/2/11

1. Introduction

Current studies, design methodology, project management, and information technology all benefit from the High-Performance Computing (HPC) environment's centralized management of several distributed high-performance computers from a range of businesses. One of the most important aspects of a HPC system is the ability to run simultaneous workloads on many processors. A crucial factor in the performance of parallel applications on HPC systems is how jobs are distributed between the available processors [8].

The necessity of inter-processor communication hinders the execution of parallel applications in HPC systems. When data is sent between activities on multiple processors, there is an additional overhead. Using HPC systems featuring diverse computers increases the need for creating high-quality job schedules. An additional factor that needs to be considered when developing a scheduling algorithm is how long a job will take to complete while running on different processors. Task scheduling has become increasingly critical as HPC and heterogeneous clusters have grown in size [9].

The performance, where the First-In-First-Out (FIFO) method, performance setting up process, and reasonable arrangement process are the job setting up systems with high performance models [3]. As a result, there's an excessive amount of wasted resources and a considerable delay in task response time when these resource scheduling strategies are implemented. Because of this, the algorithm tends to slip into so-called "local optimality." A general model of scheduler structure is depicted in Figure 1.

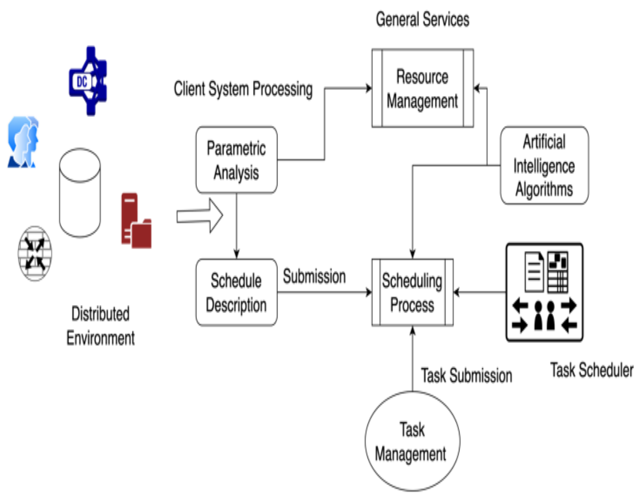

Figure 1. General Structure of scheduler model.

Also, the issue of inefficient use of available resources is a primary concern [1]. A cluster management system is critical in a high-performance computer cluster. Cluster management relies heavily on scheduling. To schedule a job, the entire process from submission to completion and the cluster's many pieces must be taken into consideration [16]. We can take full advantage of high performing computation cluster system services and guarantee that the process is quick and efficient by building and using an effective task scheduling method. The task-resource connection is what we mean by "task scheduling." As a result of extracting and evaluating the system management tasks, it is possible to enhance the task scheduling performance in direct relation to the load characteristics [2].

The goal of a task scheduler is to better use computer resources, reduce overall task execution time, and increase user satisfaction by allocating running tasks to the most appropriate processing nodes. Combinatorial optimisation is a special application of task scheduling, which becomes Non-deterministic Polynomial-time (NP)-hard as the cluster size grows. It's becoming increasingly common, though, to use an intelligent algorithm to solve these kinds of difficulties. Artificial fish swarm algorithms are used to enhance task scheduling performance, for example, when node execution ability and overall task execution time are considered optimisation goals for nodes in swarms [20].

Since cluster expansion is inevitable, past and present hardware will have to be replaced, which will generate internal heterogeneity issues, Task scheduling techniques are now focused on heterogeneous settings [12]. In some cases, network topology heterogeneity can be caused by differences in the models of the Central Processing Unit (CPU) and memory on individual nodes, or by the addition or deletion of hardware. Task scheduling techniques for diverse contexts have therefore been the subject of several academic studies and advances. If the work execution planning strategy is saved using the Length Approximation Timer Ending (LATE) algorithm, then the task library's fastest and easiest tasks are prioritised for execution [17].

The LATE algorithm is a comment method for dealing with the issue of speculative execution. LATE scheduling technique based on resource prediction, for example, has been studied extensively since then by a slew of academics. Because the LATE approach does not address the issue of optimising data locality, the issue of reading data from several systems has been resolved. The literature, on the other hand, makes use of the results of previously performed work [7].

To keep track of the present cluster's operational status and alter the job assignment strategy in a timely manner. Although this method is quick and easy to use, it is tough to get greater outcomes with it. The work scheduling method can be improved by using a swarm intelligence algorithm [15]. In order to better understand and optimize the routine of work arrangement procedures in a high-performance setting, literature provides the consequences of experiments with various intelligent algorithms. Using an intelligent approach, the researchers were able to demonstrate that task execution time may be reduced while also improving the scheduling effect [16].

2. Existing Work

Computational intelligence techniques may be used to schedule tasks in HPC environments, and we'll discuss these methods in detail in this section. Real-time approaches are known as the methodology employed in these computing environments [5]. Once the user's request parameters and resources have been validated and verified, the client creates an order to improve reliability Job submission description language for the essential facility dispensation scheme, which then processes the request for service. After receiving a user's task submission demand, the core service processing system will instantly begin the task scheduling module [11].

Responding to a manipulator's demand for facility description, kind, and computing duration, it determines a list of presently accessible resources. Then, after finishing the transformation and development of the job proposal, run the job plan on an HPC using the shortest queue time for the resource scheduling job. This approach is clear and straightforward to put into practice. Simultaneous dispensation in the structure's essential facilities is in great demand when the task assignment requests are significantly filled. It is also constrained by the greatest variety of network connections that may be made between dispersed modules [6].

Due to the increasing complexity of job scheduling issues, new intelligence algorithms, including evolutionary algorithms, simulated annealing, and taboo search, have been developed in recent years. Using a standard task scheduling method in a significant computational environment is insufficient: ignore the quality of service while focusing on efficiency; focus on justice while reducing efficiency [14]. High performance computing's programming framework has an intelligent scheduling mechanism. Both the total completion time and average completion time are reduced when improved task scheduling is used.

The results of a simulation experiment comparing real-time task scheduling with intelligent scheduling show that the latter is more efficient in a HPC environment [19]. As a result of extensive study into computer technology and a scheduling challenge, researchers have proposed a strategy constructed on the structure of an optimally efficient task allocation algorithm and a greedy algorithm. The standard task scheduling method emphasises efficiency, but the recently suggested approach emphasises service quality and achieves, for the first time, a double equilibrium in job alignment in HPC settings [4].

As an example, in computer science, the swarm intelligent scheduling algorithm is one specific approach for addressing computing difficulties that may be simplified to optimal pathways across graphs. In swarm intelligence approaches, these algorithms belong to the ant colony algorithm family and represent certain heuristic optimisations [18]. A broader range of mathematical problems may now be solved using the original concept, which has now evolved to include new problems based on various elements of ant behaviour. ACO uses a model-based search and has some resemblances to algorithms for estimating distributions [12].

In a high-performance computer environment, the question of how to properly schedule jobs is critical. An ant colony algorithm and reinforcement learning-based cooperative task scheduling approach are developed in light of the fact that resource allocation is an NP-hard issue, and the current task resource allocation technique has long scheduling times and unbalances system burden [10]. Initially, the ant system was used to unravel the visiting trip issue, with the aim of discovering the unswerving distance between two points on a given route. The basic algorithm is built on a collection of ants, each of which completes one of the potential circuits across the city [16].

Every time the ant moves between cities, it does so according to some set of laws. Each city may only be visited once; therefore, a faraway one has a lower probability of getting picked. An edge between two cities with a more intense pheromone trail has a better chance of being picked if the journey is short; if the journey is long, the ant deposits more pheromones on all edges it has crossed During each repeat, the pheromone trails go away [13].

3. Proposed System

Assume a HPC cluster is in place. Hence, task scheduling inside a HPC architecture is all about finding the shortest overall path of resource allocation. This can be observed in an HPC setting, where the quality of a job scheduling algorithm is increasingly measured by the amount of time it takes to complete all of its jobs. This present job line and job set show separate jobs, and each task might still execute on one thread component.

Environmental queues are introduced in an intelligence-optimised job scheduling paradigm. The resource approximator, resource organiser, and resource gatherer are the three main components of the core module. There is a library of application and user mapping information in the front-end service, which collects queue information from the HPC properties on a regular basis; this information includes the queue name and status as well as the approximated hubs that can be used. On the root of an HPC line, computing resources are defined. Prioritising duties on an environmental level and operating on a particular user are both conducted to rectify the collected data in an automated fashion.

Task queries that cannot obtain high computing properties and tasks that surpass the handler's bounds are added to the task queue by the front-end scheduler. When it comes to front-end scheduling, FIFO principles guide processing these job queues and responding to task requests. If the work is completed, remove it from the front task tracker; otherwise, it will remain in the queue until it is completed. The work status is refined after the introduction of an intelligent scheduling algorithm in the HPC system, making administration, operation, and maintenance easier. The user, on the other hand, must maintain a clear display of current task progress. Consequently, operational status is split into user and system status. Figure 2 shows the proposed system architecture.

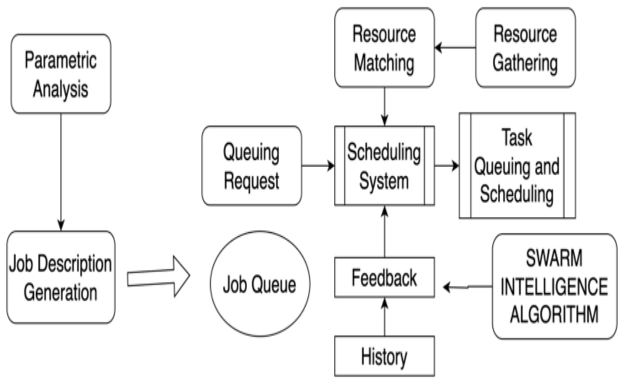

Figure 2. Proposed system architecture.

The user's perspective displays that the computer is uploading files and that the task request has been approved and is being queued or scheduled in the primary job queue. A failed operation denotes either a normal or anomalous termination of a job that is executing in a HPC environment. New is the state of the task request when it first enters the core queues; delay is the state when it first appears in the front-end queues; and scheduling indicates that the scheduler is working on it. Scheduled designates that the scheduling flops, and final is the state when irregular handler information records are sent over high-performance servers.

In this way, the procedure is analogous to an ant creeping over a graph G. Every vertex on G represents a task, and the crawling stops after all chores have been completed. Each job may only be assigned once, and the value of each decision-making variable has particular restrictions. Solution S' expected execution time is an objective function, and the algorithm's purpose is to discover a solution that reduces the anticipated execution time to the minimum feasible value.

Every conceivable choice has a signal pertaining to it, which is initialised with the commencement procedure and becomes efficient as the procedure progresses. In a certain assignment procedure, each assessment of the flexible choice is linked to the earlier process. The minimum obtained with this approach is computed based on the variety of jobs, the duration of the tasks, and the processing power of the resources because of the active and diverse nature of supply collection in HPC.

4. Results and Discussion

Particle swarm and ant colony algorithms are two of the most commonly used swarm intelligence algorithms, and they both imitate real bird and ant colony behaviour. There are certain drawbacks to using swarm intelligence algorithms, which mimic the natural social behaviour of animals. Many parameters must be configured for the ant colony algorithm to work. A substantial percentage of exploratory arbitrary motions are already required when the data item is scooped up or laid down in the ant colony clustering method. In addition, its input parameters are extremely sensitive to small changes in values. The memory configuration of nodes is listed in Table 1.

Table 1. Memory configuration of nodes.

Node memory	Number of nodes	CPU memory
8	52	45
16	124	74
32	10	154
64	189	124
128	85	11

A new version of the proposed approach is proposed in this paper in order to make better use of available resources, reduce overall job execution time, and increase user satisfaction. Simulated trials in a high-performance computer environment were used to verify the efficacy of the suggested technique. A four-node HPC cluster was used for the experiments, each of which provided a task function. One administration node, 13 submission nodes, and 316 computing nodes make up the HPC system (execution nodes). In addition, the system includes multiple sequencers, storage servers, and other components that are primarily connected by 10-gigabit Ethernet.

This SMP-designed server is used by compute nodes, and Advanced Micro Device (AMD) 64-bit processors are primarily used. 2600 MHZ is the most common CPU clock speed, while 24 to 30 logical CPUs are the most common node numbers. Storage nodes are used as the primary data storage in an HPC system. Resource management and job scheduling are carried out via the operation management system. A client-side submission is required. The compute node's CPU core count and memory usage are displayed. The nodes' hardware and operating system setups are listed. This experiment uses the same parameter settings throughout to ensure that the results are comparable. Node memory analysis is depicted in Figure 3.

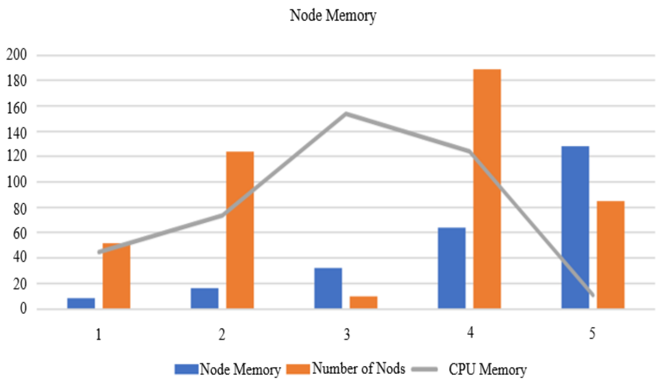

Figure 3. Node memory analysis.

Table 2 shows the performance of the proposed system. Numerous HPC activities demand substantial resources and time to be run successfully. For instance, a multi-threaded operation may demand as much as eight 4-core nodes, 8.5 GB of peak RAM, and 1.5 terabytes of storage space to complete a single test. It may take a week or so to complete the entire process. Because of this, we find it important to thoroughly comprehend the features and arrangement features of tasks and systems in order to improve and optimise their schedules and actions. All measurable tests for the proposed approach are carried out to correctly and fairly examine the performance of the scheduling algorithm.

Table 2. Performance of proposed system.

	Average of finishing	
Task	General scheduling	Proposed scheduling
0	10	8
25	15.258	8.225
50	20.516	9.249
75	25.774	9.273
100	31.032	9.297
125	36.29	9.321
150	41.548	9.345
175	46.806	9.369
200	52.064	9.393
225	57.322	9.417
250	62.58	9.441
275	67.838	9.465
300	73.096	9.489
325	78.354	9.513
350	83.612	9.537
375	88.87	9.561
400	94.128	9.585
425	99.386	9.609
450	104.644	9.633
475	109.902	9.657
500	115.16	9.681

In the first experiment, a job was delivered to the HPC every second throughout the scheduling time window of one second. All of these tasks were given varying degrees of computational difficulty at random (about 100 points). Two batches of 2000 jobs each are sent to the HPC for processing. To begin, our swarm intelligence scheduling algorithms scheduled 1000 jobs for the HPC. It was only after the first thousand jobs had been completed that we began submitting the second thousand. Scheduling time analysis of jobs has been depicted in Figure 4.

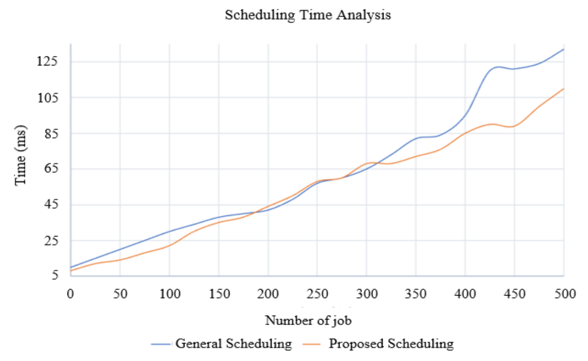

Figure 4. Scheduling time analysis.

This graph compares the swarm intelligence scheduling algorithm's average execution time to that of the arbitrary arrangement approach. That is observed over booking. Organisations employing intelligent arrangement procedures can effectively and competently schedule service duties. It takes less than 0.3 seconds to accomplish nearly 500 of the 1000 jobs, and less than 0.5 seconds to complete more than 800 of them. The error rate analysis is depicted in Figure 5.

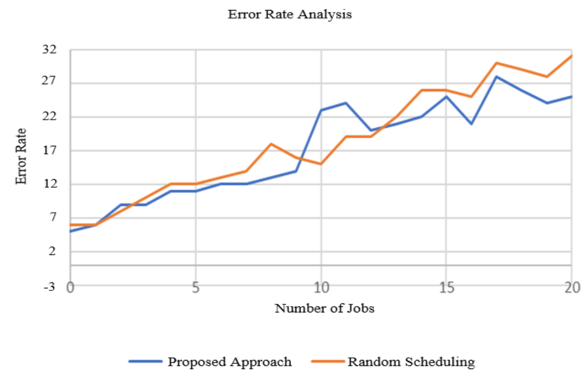

Figure 5. Error rate analysis.

Everyone knows that the middleware in the system is responsible for determining which HPC resources are most suited to the tasks that users submit. Some task submission requests fail due to several internet connections under large concurrent task submission requests as transmission increases and the number of task assignment requests significantly grows.

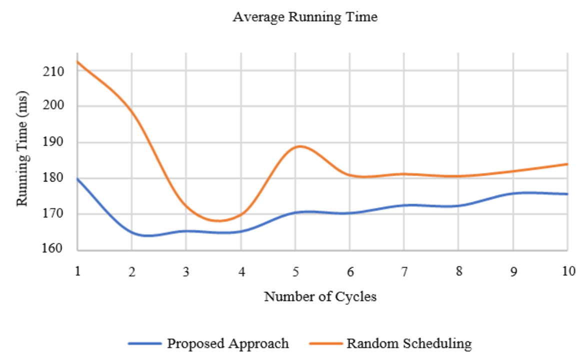

Figure 6. Average running time.

In Figure 6, average running time of the proposed system is shown. The former is clearly faster than the latter. Swarm intelligence scheduling was used to accomplish over 800 jobs on the HPC, whereas random scheduling only managed to complete 400. As a result, comparing and analysing the prediction's inaccuracy will be essential. The flat federation indicates the prophecy fault, while the perpendicular federation reflects the jobs in our experiment. The variation was 5.95, and the mean of the predicted errors was 3.8 seconds. It's worth noting that the real execution time might have been several dozen seconds. Because of this, the 4-second forecast error was acceptable. A sampled analysis of the average running time is listed in Table 3.

Table 3. Analysis of average running time.

Cycles	Average running time
1	180
2	179.72
3	179.575
4	179.43
5	179.285
6	179.14
7	178.995
8	178.85
9	178.705
10	178.56

We performed a total of 20 experiments to arrive at the final average. In each test, many jobs might arrive at the HPC at the same time. That number of tasks increases from 40 to 100. Tasks planned using average and random scheduling were measured for average execution time. Swarm-based scheduling has the greatest results, as can be seen in the graphs. When they scheduled 80 jobs, the average execution time of our approach was around 20 seconds longer than the average execution time of the other method. The gulf in time was enormous. Convergence time analysis between various algorithms is shown in Figure 7.

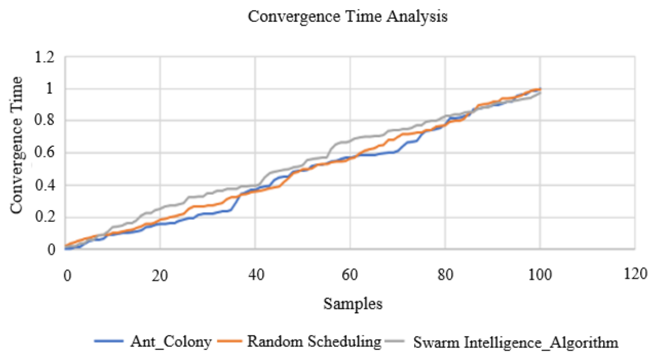

Figure 7. Convergence time analysis.

In this case, we can see how the computational intelligence algorithm is coming to a point of convergence. Near-optimal allocation can be found to converge in roughly 50 iterations, according to the research. Our suggested scheduling technique outperforms the standard job scheduling approach in simulations. Task waiting times can be reduced, system throughput increased, task response times improved, and system resources better used with a task scheduling method based on evolutionary algorithms.

It is common that the system middleware matches user-submitted tasks with the best HPC resources available in the system. Some task submission requests fail due to excessive network connections under large concurrent task submission requests as transmission and the number of task assignment requests both significantly rise. With the help of predictive maintenance, expenditures may be kept to a minimum while uptime is maximised and the reliability of the system is protected from potential failures. The use of neural networks and deep learning offers a promising approach for this application domain, as shown by the outcomes of machine learning models for event prediction in time series and the continuous monitoring of mission essential equipment.

The implementation issues in these situations emerge from the system's underlying hardware, yet overcoming them is crucial for the effective deployment of these concepts. To overcome these obstacles, there must be consensus among stakeholders to support joint efforts to create effective business models that benefit all parties. Using random forest classifiers, we compare resource utilisation data from two settings using the F1-score metric, and we find that the optimal value is 0.97. From the data collected by monitoring computer nodes, a prediction model is developed using descriptive statistics and supervised machine learning techniques. While it did achieve up to 79% failure detection rates during validation in a production context, this technique also showed a large number of false positives.

A reliable power supply is essential in mission-critical supercomputing settings, since it lessens the chances of service interruption and, in turn, lowers operating expenses. By collecting and analysing system logs, compute node failures in a HPC system may be predicted through the development of classification algorithms for data mining. A computational node's consumption and error records are combined in this technique. The HPC system uses a prediction model to determine if the breakdown will happen subsequently.

5. Conclusions

According to the findings of this work, we have suggested an intelligent scheduling model for numerous services using the proposed approach in HPC. An enhanced swarm intelligence system has been created that uses projected outcomes to plan tasks. Task completion times can be reduced by as much as 50% using the swarm intelligence algorithm, which is implemented in our main scheduling module. There is evidence that the suggested approach is superior to the typical task scheduling method in terms of performance. Therefore, the workflow scheduling system based on the proposed approach would efficiently decrease the waiting time, enhance processes, productivity, responsiveness, and organisation supply exploitation with improved impact.

References

[1] Amer D., Attiya G., Zeidan I., and Nasr A., "Elite Learning Harris Hawks Optimizer for Multi-Objective Task Scheduling in Cloud Computing," *The Journal of Supercomputing*, vol. 78, pp. 2793-2818, 2022. https://doi.org/10.1007/s11227-021-03977-0

[2] Anzt H., Cojean T., Flegar G., and Göbel F., "Ginkgo: A Modern Linear Operator Algebra Framework for High Performance Computing," *ACM Transactions on Mathematical Software*, vol. 48, no. 1, pp. 1-33, 2022. DOI:10.1145/3480935

[3] Bartolini A., Borghesi A., Lombardi M., Milano M., and Benini L., "Anomaly Detection Using Autoencoders in High Performance Computing Systems," *in Proceedings the of 31st AAAI Conference on Innovative Applications of Artificial Intelligence*, Hawaii, pp. 9428-9433, 2019. https://doi.org/10.1609/aaai.v33i01.33019428

[4] Borghesi A., Libri A., Benini L., and Bartolini A., "Online Anomaly Detection in HPC Systems," *in Proceedings of the IEEE International Conference on Artificial Intelligence Circuits and Systems*, Hsinchu, pp. 229-233, 2019. https://ieeexplore.ieee.org/document/8771527

[5] Carvalho T., Soares F., Vita R., Francisco R., Basto J., and Alcalá S., "A Systematic Literature Review of Machine Learning Methods Applied to Predictive Maintenance," *Computers and Industrial Engineering*, vol. 137, pp. 106024, 2019. https://doi.org/10.1016/j.cie.2019.106024

[6] Essien A. and Giannetti C., " A Deep Learning Model for Smart Manufacturing Using Convolutional LSTM Neural Network Autoencoders," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6069-6078, 2020. DOI:10.1109/TII.2020.2967556

[7] Gupta S., Iyer S., Agarwal G., and Manoharan P., "Efficient Prioritization and Processor Selection Schemes for HEFT Algorithm: A Makespan Optimizer for Task Scheduling in Cloud Environment," *Electronics*, vol. 11, no. 16, pp. 1-15, 2022. https://doi.org/10.3390/electronics11162557

[8] Jamil B., Ijaz H., Shojafar M., Munir K., and Buyya R., "Resource Allocation and Task Scheduling in Fog Computing and Internet of Everything Environments: A Taxonomy, Review, and Future Directions," *ACM Computing Surveys*, vol. 54, no. 11s, pp. 1-38, 2022. https://doi.org/10.1145/3513002

[9] Jena B., Nayak G., and Saxena S., *High-Performance Medical Image Processing*, Apple Academic Press, 2022. DOI:10.1201/9781003190011-12

[10] Kruekaew B. and Kimpan W., "Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm with Reinforcement Learning," *IEEE Access*, vol. 10, pp. 17803-17818, 2022. https://ieeexplore.ieee.org/document/9708723

[11] Li C., Zhang C., Ma B., and Luo Y., "Efficient Multi-Attribute Precedence-based Task Scheduling for Edge Computing in Geo-Distributed Cloud Environment," *Knowledge and Information Systems*, vol. 64, pp. 175-205, 2022. https://link.springer.com/article/10.1007/s10115-021-01627-8

[12] Nayak S., Parida S., Tripathy C., and Pattnaik P., "An Enhanced Deadline Constraint-based Task Scheduling Mechanism for Cloud Environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 2, pp. 282-294, 2022. https://doi.org/10.1016/j.jksuci.2018.10.009

[13] Pirozmand P., Javadpour A., Nazarian H., Pinto P., Mirkamali S., and Ja'fari F., "GSAGA: A Hybrid Algorithm for Task Scheduling in Cloud Infrastructure," *The Journal of Supercomputing*, vol. 78, no. 4, pp. 17423-17449, 2022.. https://doi.org/10.1007/s11227-022-04539-8

[14] Sellami B., Hakiri A., Yahia S., and Berthou P., "Energy-Aware Task Scheduling and Offloading Using Deep Reinforcement Learning in SDN-Enabled IoT Network," *Computer Networks*, vol. 210, pp. 108957, 2022. https://laas.hal.science/hal-03648574/document

[15] Shukla A., Kumar S., and Singh H., "Fault Tolerance-Based Load Balancing Approach for Web Resources in Cloud Environment," *The International Arab Journal of Information Technology*, vol. 17, no. 2, pp. 225-232, 2020. https://www.iajit.org/portal/PDF/Vol%2017,%20No.%202/17514.pdf

[16] Talaat F., Ali H., Saraya M., and Saleh A., "Effective Scheduling Algorithm for Load Balancing in Fog Environment Using CNN and MPSO," *Knowledge and Information Systems*, vol. 64, no. 3, pp. 773-797, 2022. https://doi.org/10.1007/s10115-021-01649-2

[17] Tripathi G. and Kumar R., "A Heuristic-Based Task Scheduling Policy for QoS Improvement in Cloud," *International Journal of Cloud Applications and Computing*, vol. 12, no. 1, pp. 1-22, 2022. DOI:10.4018/IJCAC.295238

[18] Wang X., Wang C., Bai M., Ma Q., and Li G., "HTD: Heterogeneous Throughput-Driven Task Scheduling Algorithm in MapReduce," *Distributed and Parallel Databases*, vol. 40, no. 1, pp. 135-163, 2022. https://doi.org/10.1007/s10619-021-07375-6

[19] Yadav A., Tripathi K., and Sharma S., "An Enhanced Multi-Objective Fireworks Algorithm for Task Scheduling in Fog Computing Environment," *Cluster Computing*, vol. 25, no. 2, pp. 983-998, 2022. https://doi.org/10.1007/s10586-021-03481-3

[20] Yurek O. and Birant D., "Remaining Useful Life Estimation for Predictive Maintenance Using Feature Engineering," *in Proceedings of the Innovations in Intelligent Systems and Applications Conference*, Izmir, pp. 1-5, 2019. DOI:10.1109/ASYU48272.2019.8946397

Srinivasan Narayanasamy works as Professor in CSE Department of Rajalakshmi Engineering College, Chennai, Tamilnadu, India. He has more than 25 years of teaching experience, and his areas of specialization include Software Engineeering, Machine Learning and Network Engineering.

MohanKumar Palanichamy works as Professor in CSE Department of Hindustan Institute of Technology and Science, Chennai, Tamilnadu, India. He has more than 20 years of teaching experience, and his areas of specialization include Artificial Intelligence, Data Analytics and Machine Learning.

Selvam Lakshmanan works as Associate Professor in CSE Department of Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India. He has more than 23 years of teaching experience, and his areas of specialization include Cloud Computing, Cryptography and Network Security.

Selvam Lakshmanan works as Associate Professor in CSE Department of Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India. He has more than 23 years of teaching experience, and his areas of specialization include Cloud Computing, Cryptography and Network Security.

ArokiaRenjit Jerald works as Professor in CSE Department of Jeppiaar Engineering College, Chennai, Tamilnadu, India. He has more than 20 years of teaching experience, and his areas of specialization include Intrusion Detection Systems, Network security and Machine Learning.