

Mean makespan task scheduling approach for the edge computing environment

Nisha Saini, Jitender Kumar

Department of Computer Science and Engineering, Deenbandhu Chhotu Ram University of Science and Technology, Murthal (Sonipat), India

Article Info

Article history:

Received Feb 7, 2024

Revised Apr 4, 2024

Accepted Apr 16, 2024

Keywords:

Edge computing

Makespan minimization

Processing time optimization

Task allocation

Task scheduling

Virtual machines

ABSTRACT

Task scheduling in the edge computing environment poses significant challenges due to its inherent NP-hard nature. Several researchers concentrated on minimizing simple makespan, disregarding the reduction of the mean time to complete all tasks, resulting in uneven distributions of mean completion times. To address this issue, this study proposes a novel mean makespan task scheduling strategy (MMTSS) to minimize simple and mean makespan. MMTSS optimizes the utilization of virtual machine capacity and uses the mean makespan optimization to minimize the processing time of tasks. In addition, it reduces imbalance by evenly distributing tasks among virtual machines, which makes it easier to schedule batches subsequently. Using genetic algorithm optimization, MMTSS effectively lowers processing time and mean makespan, offering a viable approach for effective task scheduling in the edge computing environment. The simulation results, obtained using cloudlets ranging from 500 to 2000, explicitly demonstrate the improved performance of our approach in terms of both simple and mean makespan metrics.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Nisha Saini

Department of Computer Science and Engineering, Deenbandhu Chhotu Ram University of Science and Technology

Murthal-131039, Sonipat, India

Email: saini.nisha0203@gmail.com

1. INTRODUCTION

The extensive deployment of internet of things (IoT) applications has led to the development of edge computing, offering low latency and fast response times for time-sensitive applications, including healthcare, emergency services, and traffic monitoring. The edge layer integrates edge and cloud resources to offer services and streamline data flow management [1]. The diversity of edge computing necessitates the use of efficient techniques to effectively optimize user requests or workloads. Load balancing is essential in edge computing to handle the growing number of users and effectively handle all user requests [2]. Adequate load balancing is crucial for improving resource utilization [3], minimizing makespan [4], and optimizing the overall performance of edge computing systems [5]. Two approaches for evenly distributing cloud load monitoring are virtual machines (VMs) and task scheduling. In edge computing, task scheduling is considered an NP-hard problem [6] due to the diverse configurations of hosts and VMs, which can quickly adapt to fluctuating user requests. It is very challenging to identify every prospective mapping among tasks and resources in the edge computing paradigm. Consequently, there is an imperative requirement for an efficient task-scheduling technique that prioritizes the strategic allocation of tasks. The goal is to prevent any VM from being underloaded or overloaded, ensuring an evenly distributed workload across all VMs.

Existing heuristics and machine learning approaches designed for task scheduling in edge computing have issues with limited relevance and quick adaptability, which hinder their capacity to solve problems optimally [7]–[10]. Furthermore, these techniques have challenges in identifying temporal workload patterns and are predominantly suitable for centralized deployments. To address these limitations, academics and researchers have diligently investigated several methods to find the optimal solutions in task scheduling [11]–[14]. Shukri *et al.* [15] developed the enhanced multi-verse optimizer (EMVO) to enhance task-scheduling efficiency. They evaluated the impact of EMVO on makespan time, throughput, and resource utilization. EMVO achieved a reduced makespan and improved throughput and resource utilization compared to the original multi-verse optimizer (MVO) and particle swarm optimization (PSO). However, the performance of EMVO degraded the system performance due to an increase in makespan time. Turkeš *et al.* [16] performed a meta-analysis on the adaptive layer in adaptive large neighborhood search (ALNS) and identified that it could enhance performance by 0.14%. The meta-analysis studies differ in several characteristics that may impact the significance of the adaptive layer.

Sun *et al.* [17] emphasized reducing the makespan in workflow scheduling within complex networks in edge computing. They introduced the improved greedy search (IGS) and improved composite heuristic (ICH) methods, surpassing current scheduling algorithms. This technique had a restricted energy capacity for each device and was interdependent among processing tasks. Li *et al.* [18] introduced a mixed-integer linear programming (MILP) model for task scheduling in a toy-edge-cloud architecture, resulting in notable reductions in solution time. Zhang [19] focused on the concurrent connection between tasks and applied the ant colony method to improve task parallelism and decrease task delays. Although this approach revealed significant global optimization performance, it had a poor convergence pace. Agarwal *et al.* [20] presented a cloud-distributed scheduling model that considers multiple optimization objectives such as makespan, cost, energy consumption, resource utilization, and load. They designed an expert advisor (EA) using the sine function to achieve the best solutions for this framework. However, despite these algorithms offering benefits in accomplishing optimal solutions with reduced makespan and energy consumption compared to existing approaches, they have limitations in lacking a thorough quantitative examination of the benefits of makespan minimization compared to conventional techniques.

To address these limitations, researchers began to explore the combination and improvement of metaheuristic algorithms. For instance, Vispute and Vashisht [21] devised a hybrid algorithm combining genetic algorithm (GA) and particle swarm optimization (PSO) to minimize energy consumption in user-terminal devices. Mangalampalli *et al.* [22] presented a heuristic algorithm called GAWOA that utilized genetic and whale optimization algorithms to reduce the total cost. In the context of comprehensive resource scheduling, An *et al.* [23] introduced an improved reference vector-guided evolutionary algorithm that employed a standard distribution angle penalty distance strategy to select the optimal solution, considering factors such as makespan, cost, load, user expectation, and task completion rate. Wang *et al.* [24] introduced a two-stage joint optimization model that deals with the flexible job shop scheduling issue and preventive maintenance. This approach helped to save costs in dynamic production environments with frequent product changes. The proposed model has shown effectiveness, but its scalability and broader applicability in other industrial situations need further investigation. The multi-objective crow search algorithm (CSAMOMC) devised by Akraminejad *et al.* [25] efficiently reduced costs and makespan in scientific cloud operations. It outperforms heterogeneous earliest finish time (HEFT) and time-cost compromise (TC3pop) by an average of 4.42% and 4.77%, respectively. Despite the favorable results, the study predominantly relies on a comparative examination. It fails to meticulously investigate the wider environmental ramifications or contemplate compromises in other facets, such as energy efficiency.

The literature evaluation emphasized the significance of conducting in-depth quantitative evaluations to determine the effectiveness of novel task-scheduling optimization algorithms in reducing makespan compared to traditional methods. The existing literature primarily focuses on minimizing simple makespan through various strategies and algorithms, without addressing the current resource utilization and workload of edge servers. This omission may result in uneven distribution of workloads and inefficient resource utilization, primarily due to the dynamic nature of the computing resources in the edge environments. Based on previous research, we examined and devised a methodology to reduce the simple makespan and processing time by minimizing the mean makespan of individual tasks across all VMS. We designed our approach to establish a task scheduling strategy using the GA, which optimizes parameters such as simple makespan, processing time, optimal resource utilization, and imbalance across VMs by minimizing the mean makespan of individual tasks across all VMs. We have organized this paper as follows: Section 2 explains the proposed research method, while section 3 discusses the findings of this study. Section 4 concludes the paper. To provide readers with a comprehensive grasp of the organization of the work, Table 1. presents a compilation of acronyms that will frequently appear throughout this study.

Table 1. List of acronyms

Symbol	Description
NP-hard	Non-deterministic polynomial time
MMTSS	Mean makespan task scheduling strategy
IoT	Internet of things
VMs	Virtual machines
EMVO	Enhanced multi-verse optimizer
ALNS	Adaptive large neighborhood search
IGS	Improved greedy search
ICH	Improved composite heuristic
MILP	Mixed-integer linear programming
EA	Expert advisor
GA	Genetic algorithm
PSO	Particle swarm optimization
GAWOA	Genetic algorithm and whale optimization algorithm
CSAMOMC	Multiobjective crow search algorithm
HEFT	Heterogeneous earliest finish time
TC3pop	Time-cost compromised
AET	Average execution time

2. PROPOSED METHOD

The proposed work addresses the challenge of minimizing makespan in edge computing. The main objective of this study is to maximize the efficient utilization of VM capacity in the edge environment. The primary intent is to decrease the simple makespan by strategically allocating tasks among several VMs. The proposed work involves optimizing resource utilization and maintaining a uniform workload distribution. Subsection 2.1 describes the problem formulation for achieving the objective of this study. Subsequently, Subsection 2.2 outlines the proposed strategy.

2.1. Problem formulation

The primary goal of this study is to reduce the simple makespan by efficiently utilizing the VM capacity. This optimization ensures efficient VM utilization and offers additional benefits such as reduced processing times, simple makespan, low imbalance on each VM, simplified batch task scheduling in subsequent iterations, and improved accuracy compared to directly reducing makespan. In edge computing, task scheduling involves allocating tasks to the most suitable VMs in data centres to meet user requirements. Consider a set of n tasks $T = \{T_1, T_2, T_3, \dots, T_n\}$, and a set of m virtual machines $VM = \{M_1, M_2, M_3, \dots, M_m\}$. Let P_{ij} be the processing time of task i on the virtual machine M_j and C_{ij} be the completion time of task i when assigned to the virtual machine M_j . A binary decision variable x_{ij} representing whether task T_i is assigned to the virtual machine M_j , with a value of 1 indicating assignment and 0 indicating no assignment. Each task is assigned to each VM, aiming to minimize the fitness of the objective function. Consequently, the proposed strategy minimizes the simple makespan by reducing the mean makespan across all VMs. The formulation of the objective and associated constraints is defined as (1).

$$\text{minimize } \frac{1}{m} \sum_{j=1}^m C_{max}^{(j)} \quad (1)$$

$$\text{subject to } C_{ij} = \frac{1}{m} \sum_{j=1}^m P_{ij} \quad (1a)$$

$$C_{max} = \max_{1 \leq i \leq n} \{ C_{ij} \mid 1 \leq j \leq m \} \quad (1b)$$

$$\sum_{j=1}^m x_{ij} = 1, \quad \forall i \in \{1, 2, 3, \dots, n\} \quad (1c)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in \{1, 2, 3, \dots, n\}, \forall j \in \{1, 2, 3, \dots, m\} \quad (1d)$$

Constraints (1a) indicate that each task is assigned to precisely one VM. Constraint (1b) calculates the simple makespan as the maximum completion time across all tasks and VMs. Constraint (1c) ensures that each task is assigned to precisely one VM, and constraint (1d) defines that the decision variable x_{ij} are binary.

2.2. Mean makespan task scheduling strategy

This section outlines a strategic approach for minimizing the simple makespan in an edge computing environment. Our strategy reduces the mean makespan, which assists in addressing issues related to the

uneven distribution of the mean completion time of tasks. Every random variable approaches the mean and provides a better path for optimal solutions. Our technique stipulates the most effectively utilized elucidation for this problem.

Consider an assortment of n tasks and m virtual machines, where each task is evenly distributed across the VMs. The completion time C_{ij} of the task T_i on each virtual machine M_j can be represented as (2).

$$C_{ij} = \frac{1}{m} \sum_{j=1}^m P_{ij} \quad (2)$$

where P_{ij} is the processing time of the task T_i on virtual machine M_j . Makespan C_{max} is the maximum completion time for all tasks. It can be expressed as (3).

$$C_{max} = \max_{1 \leq i \leq n} \{ C_{ij} \mid 1 \leq j \leq m \} \quad (3)$$

The mean makespan \bar{C}_j is the mean completion time for all tasks. It can be expressed as (4).

$$\bar{C}_j = \frac{1}{n} \sum_{i=1}^n C_{ij} \quad (4)$$

On substituting the value of completion time into the mean makespan expression, we get (5) to (7).

$$\bar{C}_j = \frac{1}{n} \sum_{i=1}^n \left(\frac{1}{m} \sum_{j=1}^m P_{ij} \right) \quad (5)$$

$$\bar{C}_j = \frac{1}{mn} \sum_{i=1}^n \sum_{j=1}^m P_{ij} \quad (6)$$

$$\bar{C}_j = \frac{1}{mn} \sum_{j=1}^m \sum_{i=1}^n P_{ij} \quad (7)$$

Since each task is evenly allocated to each VM, the values of processing time P_{ij} remains constant $\forall i$ and j . Therefore, this expression can be rewritten as (8) and (9).

$$\bar{C}_j = \frac{1}{mn} (mn)(P_{ij}) \quad (8)$$

$$\bar{C}_j = P_{ij} \quad (9)$$

This demonstrates that the mean makespan is directly proportional to the processing time of tasks on the VM. Thus, by lowering the mean makespan, the processing time and completion time for all tasks on all VMs will also be minimized. As the simple makespan C_{max} represents the maximum completion time across all VMs, reducing the completion time \bar{C}_j of any task on any VM will correspondingly decrease the C_{max} . Therefore, achieving a minimum in the mean makespan assures a minimum in the simple makespan.

3. RESULTS AND DISCUSSION

In this section, we assess and contrast the performance of the proposed MMTSS strategy in terms of i) simple makespan and ii) mean makespan. We carried out the simulation using CloudSim [26]. We evaluated the proposed strategy using the NASA Ames iPCS/860 log dataset, which contains real-world workloads from [27] that CloudSim recognized. Parameters are discussed in Table 2. The results of our study demonstrate that our approach effectively minimizes the simple makespan and enhances the performance of individual tasks by reducing processing time. Furthermore, our strategy of minimizing the mean makespan leads to a decrease in the overall simple makespan. However, with an increase in the number of tasks, there is a higher degree of volatility. Table 3 represents a comprehensive analysis of the simple makespan (AET) and mean makespan (AET) metrics, which span a range of cloudlet quantities from 500 to 2000. This analysis aims to clarify the distinctions in the mean execution time observed across the two metrics over different numbers of cloudlets. This comparison is optimized by Figure 1, which visually depicts the experimental analysis of simple and mean makespan. It provides a visual depiction of their performance in terms of task allocation on VMs in the edge computing environment.

Table 2. Configuration of parameters in the CloudSim simulator

Entities	Parameters	Values
Cloudlets	Number of cloudlets	500-2,000
	File size	100
	Output size	100
VM	Number of VMs	6
	Bandwidth	1,000 Mbps
	RAM	512 MB
	VMM	Xen
	Number of CPU	1
	VM Image size	10,000 MB

Table 3. Comparison of simple and mean metrics

Number of cloudlets	Simple makespan (AET)	Mean makespan (AET)
500	50658.23333	93267.3
1000	105172.8333	186218.3667
1500	166021.9	296522.0667
2000	207495	403255.7333

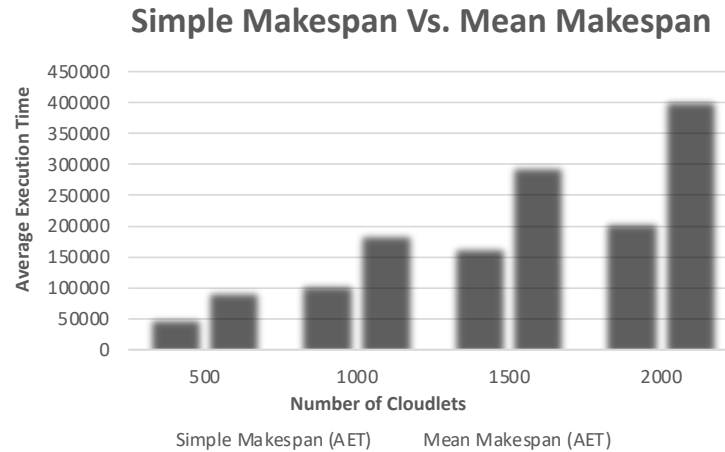


Figure 1. Comparative analysis of simple makespan and mean makespan

During our investigation, we observed an unforeseen pattern in our analysis of the correlation between the mean makespan and the measurements of the simple makespan. Generally, we expect the mean makespan values to decrease in comparison to the simple makespan, signifying an overall optimization in the scheduling process. However, the results of our study demonstrated a deviation from the predicted trend, indicating a complete reversal in the expected direction. After carefully analyzing the data, it became clear that the mean makespan readings demonstrated more significant variations than the simple makespan values. This disparity prompted an in-depth investigation of the fundamental elements that contributed to this observed pattern.

The subsequent study revealed a potential relationship between the significant variation in the mean makespan observations and the impact of outlier observations within the dataset. Outliers, which are data points that deviate significantly from many observations, have the potential to disproportionately affect the estimation of the mean. Due to their exceptional characteristics, outliers significantly impact the mean value of a dataset, which in turn increases the observed variability in the mean makespan values. Therefore, the difference between the expected and actual results highlights the significance of carefully examining the data and considering possible anomalies in statistical analysis. Let's consider a hypothetical scenario where we introduce four tasks, T1 (70 s), T2 (10 s), T3 (5 s), and T4 (15 s). These assignments have a total duration of 100 seconds, resulting in an average or mean value of 25 seconds. There are significant fluctuations, especially at extreme values like 70 seconds. Outliers have an impact on the mean, so devising a solution to address this issue is critical. Hence, developing a system that can effectively reduce the impact of outlier values and provide a more resilient optimization methodology is crucial. Despite these results, further studies will prioritize exploring alternate parameters and approaches to reduce the impact of outliers on the mean makespan computations.

4. CONCLUSION AND FUTURE SCOPE

The paper presents the MMTSS, a novel approach for minimizing makespan in the edge computing environment. This study deviates from previous literature by optimizing parameters such as simple makespan, processing time, and imbalance across VMs. This is achieved by minimizing the mean makespan of individual tasks across all VMs. Our proposed approach aims to alleviate the challenges associated with the uneven distribution of the mean completion time for tasks. The objective is to minimize tasks' simple makespan and processing time by implementing a GA technique to reduce the mean makespan across all VMs. The method enhances overall performance by optimizing processing times and lowering the simple makespan through a uniform distribution of tasks among VMs. Moreover, the paper highlights the need to effectively utilize VM capacity to ensure adequate workload allocation, minimize imbalance, and enhance task scheduling accuracy in subsequent iterations. As part of future work, we aim to incorporate additional metrics for better convergence of the metaheuristic algorithms. In addition, we also plan to extend the task scheduling strategy to the hybrid cloud environment.




REFERENCES

- [1] U. Arora and N. Singh, "IoT application modules placement in heterogeneous fog-cloud infrastructure," *International Journal of Information Technology*, vol. 13, no. 5, pp. 1975–1982, May 2021, doi: 10.1007/s41870-021-00672-4.
- [2] G. Singh, S. Prakash, and S. Kumar, "Minimizing makespan time in cloud computing using heuristic elasticity based dynamic task scheduling algorithms," *Journal of System and Management Sciences*, vol. 11, no. 2, pp. 29–47, Jun. 2021, doi: 10.33168/JSMS.2021.0203.
- [3] M. Z. Nayer et al., "LBRO: load balancing for resource optimization in edge computing," *IEEE Access*, vol. 10, pp. 97439–97449, 2022, doi: 10.1109/access.2022.3205741.
- [4] A. Pradhan and S. K. Bisoy, "A novel load balancing technique for cloud computing platform based on PSO," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 3988–3995, Jul. 2022, doi: 10.1016/j.jksuci.2020.10.016.
- [5] N. Saini and J. Kumar, "Optimizing edge computing through virtualization: a comprehensive review," *international journal of advanced research in computer science*, vol. 15, no. 1, pp. 33–44, Feb. 2024, doi: 10.26483/ijarcs.v15i1.7047.
- [6] A. A. Pavlov, E. B. Misura, O. V. Melnikov, and I. P. Mukha, "NP-hard scheduling problems in planning process automation in discrete systems of certain classes," in *Advances in Computer Science for Engineering and Education*, Springer International Publishing, 2018, pp. 429–436. doi: 10.1007/978-3-319-91008-6_43.
- [7] T. Le Duc, R. G. Leiva, P. Casari, and P.-O. Östberg, "Machine learning methods for reliable resource provisioning in edge-cloud computing: a survey," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–39, Sep. 2019, doi: 10.1145/3341145.
- [8] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020, doi: 10.1109/comst.2020.2970550.
- [9] F. Xue, Q. Hai, T. Dong, Z. Cui, and Y. Gong, "A deep reinforcement learning based hybrid algorithm for efficient resource scheduling in edge computing environment," *Information Sciences*, vol. 608, pp. 362–374, Aug. 2022, doi: 10.1016/j.ins.2022.06.078.
- [10] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: a survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2131–2165, 2021, doi: 10.1109/comst.2021.3106401.
- [11] D. Alboaneen, H. Tianfield, Y. Zhang, and B. Pranggono, "A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers," *Future Generation Computer Systems*, vol. 115, pp. 201–212, Feb. 2021, doi: 10.1016/j.future.2020.08.036.
- [12] M. Đurasević and D. Jakobović, "Heuristic and metaheuristic methods for the parallel unrelated machines scheduling problem: a survey," *Artificial Intelligence Review*, vol. 56, no. 4, pp. 3181–3289, Aug. 2022, doi: 10.1007/s10462-022-10247-9.
- [13] J. Kakkottakath Valappil Thekkepuryil, D. P. Suseelan, and P. M. Keerikkattil, "An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment," *Cluster Computing*, vol. 24, no. 3, pp. 2367–2384, Mar. 2021, doi: 10.1007/s10586-021-03269-5.
- [14] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiha, "Metaheuristic algorithms: a comprehensive review," in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, Elsevier, 2018, pp. 185–231. doi: 10.1016/b978-0-12-813314-9.00010-4.
- [15] S. E. Shukri, R. Al-Sayyed, A. Hudaib, and S. Mirjalili, "Enhanced multi-verse optimizer for task scheduling in cloud computing environments," *Expert Systems with Applications*, vol. 168, Apr. 2021, doi: 10.1016/j.eswa.2020.114230.
- [16] R. Turkeš, K. Sörensen, and L. M. Hvattum, "Meta-analysis of metaheuristics: quantifying the effect of adaptiveness in adaptive large neighborhood search," *European Journal of Operational Research*, vol. 292, no. 2, pp. 423–442, Jul. 2021, doi: 10.1016/j.ejor.2020.10.045.
- [17] J. Sun, L. Yin, M. Zou, Y. Zhang, T. Zhang, and J. Zhou, "Makespan-minimization workflow scheduling for complex networks with social groups in edge computing," *Journal of Systems Architecture*, vol. 108, Sep. 2020, doi: 10.1016/j.sysarc.2020.101799.
- [18] S. Li, W. Chen, Y. Chen, C. Chen, and Z. Zheng, "Makespan-minimized computation offloading for smart toys in edge-cloud computing," *Electronic Commerce Research and Applications*, vol. 37, Sep. 2019, doi: 10.1016/j.elerap.2019.100884.
- [19] X. Zhang, "A fine-grained task scheduling mechanism for digital economy services based on intelligent edge and cloud computing," *Journal of Cloud Computing*, vol. 12, no. 1, Mar. 2023, doi: 10.1186/s13677-023-00402-0.
- [20] G. Agarwal, S. Gupta, R. Ahuja, and A. K. Rai, "Multiprocessor task scheduling using multi-objective hybrid genetic Algorithm in Fog-cloud computing," *Knowledge-Based Systems*, vol. 272, Jul. 2023, doi: 10.1016/j.knosys.2023.110563.
- [21] S. D. Vispute and P. Vashisht, "Energy-efficient task scheduling in fog computing based on particle swarm optimization," *SN Computer Science*, vol. 4, no. 4, May 2023, doi: 10.1007/s42979-022-01639-3.
- [22] S. Mangalampalli, G. R. Karri, and U. Kose, "Multi objective trust aware task scheduling algorithm in cloud computing using whale optimization," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 2, pp. 791–809, Feb. 2023, doi: 10.1016/j.jksuci.2023.01.016.
- [23] Y. An, X. Chen, K. Gao, L. Zhang, Y. Li, and Z. Zhao, "A hybrid multi-objective evolutionary algorithm for solving an adaptive




- flexible job-shop rescheduling problem with real-time order acceptance and condition-based preventive maintenance,” *Expert Systems with Applications*, vol. 212, Feb. 2023, doi: 10.1016/j.eswa.2022.118711.
- [24] Y. Wang *et al.*, “Joint optimization of flexible job shop scheduling and preventive maintenance under high-frequency production switching,” *International Journal of Production Economics*, vol. 269, Mar. 2024, doi: 10.1016/j.ijpe.2024.109163.
- [25] R. Akraminejad, N. Khaledian, A. Nazari, and M. Voelp, “A multi-objective crow search algorithm for optimizing makespan and costs in scientific cloud workflows (CSAMOMC),” *Computing*, Feb. 2024, doi: 10.1007/s00607-024-01263-4.
- [26] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011, doi: 10.1002/spe.995.
- [27] NASA, “The NASA Ames iPSC/860 log,” *Parallel Workloads Archive*. http://www.cs.huji.ac.il/labs/parallel/workload/1_nasa_ipsc/ (accessed Feb. 07, 2024).

BIOGRAPHIES OF AUTHORS



Nisha Saini    currently immersed in her Ph.D. in computer science and engineering at Deenbandhu Chhotu Ram University of Science and Technology, Murthal, Sonipat, Haryana, India. She holds academic distinctions with B.Tech. in information technology and M.Tech. in computer science and engineering, both conferred by the same university. She possessed a three-year tenure of teaching experience prior to her current academic pursuits. Her research mainly focuses on the internet of things (IoT), cloud computing, and edge computing. She can be contacted at email: saini.nisha0203@gmail.com.



Jitender Kumar    received the M. Tech. degree in computer science and engineering from the Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India, in 2008. He has completed his Ph.D. at Deenbandhu Chhotu Ram University of Science and Technology, Murthal, Sonipat, Haryana, India. He is an assistant professor at the Department of Computer Science and Engineering, Deenbandhu Chhotu Ram University of Science and Technology. His research interests include cloud computing and mobile computing. He can be contacted at email: jitenderkumar.cse@dcrustm.org.