INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

EBTASIC: An Entropy-Based TOPSIS Algorithm for Task Scheduling in IaaS Clouds

Vivek Shrivastava[1]*, **Yasmin Shaikh**[1]

1 International Institute of Professional Studies, Devi Ahilya University, Indore-452001, Madhya Pradesh, India

OPEN ACCESS

Received: 12-04-2022

Accepted: 17-08-2022

Published: 26-09-2022

Citation: Shrivastava V, Shaikh Y (2022) EBTASIC: An Entropy-Based TOPSIS Algorithm for Task Scheduling in IaaS Clouds. Indian Journal of Science and Technology 15(37): 1850-1858. https://doi.org/10.17485/IJST/v15i37.799

* **Corresponding author**.

vivek.shrivastava@iips.edu.in

Funding: None

Competing Interests: None

Copyright: © 2022 Shrivastava & Shaikh. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: To propose an algorithm to balance resource utilization and revenue generation in the cloud environment. **Methods:** This study proposes the Entropy-Based TOPSIS algorithm for task scheduling in IaaS Clouds (EBTASIC) to balance resource utilization and revenue generation using the objective-based Entropy Weighting Method (EWM) and Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS). **Findings/Novelty:** Various performance evaluation factors are calculated using EBTASIC and compared with baseline algorithms First Come First Serve (FCFS) and Earliest Deadline First (EDF) algorithms over 12750 lease requests with hard deadlines. The actual response time of EBTASIC is 37.61 percent faster than FCFS and 47.95 percent faster than EDF. Total time spent on lease execution by EBTASIC is reduced by 3.43 percent when compared to FCFS and 3.99 percent when compared to EDF. Turnaround time of EBTASIC is lowered by 21.14 percent when compared to FCFS and 28.81 percent when compared to EDF. EBTASIC throughput is enhanced by 40.80 percent over FCFS and 54.3 percent over EDF. The average response time of EBTASIC is shortened by 9.38 percent compared to FCFS and 14.81 percent compared to EDF. Resource utilization of the proposed algorithm EBTASIC is enhanced by 25.54 percent over FCFS and 6.79 percent over the EDF algorithm.

Keywords: Task and VM Scheduling; MCDM Techniques; TOPSIS; Entropy Weighting Method; EBTASIC Algorithm

1 Introduction

Resource scheduling can be performed based on many criteria such as earliest deadline first, first come first serve, highest revenue first, shortest time first, minimum cost first, minimum makespan first, highest resource utilization first and so on. Taking into account only one criterion for scheduling will yield better results just for that criterion, but a sense of balance is essential in all parameters.

Various Multi-Criteria Decision Making (MCDM) methods are used to address decision-making challenges with multiple alternatives in VM scheduling for effective utilization of resources with optimization in other criteria [1,2].

Nayak et al. implemented the MCDM techniques: AHP, VIKOR, and TOPSIS to avoid task scheduling conflicts in backfilling algorithm for executing deadline- based tasks in cloud computing [3]. Similarly, Shariat et al. proposed the HATMOG method that was based on the smart hybrid of multiple criteria decision-making algorithms of AHP-TOPSIS and Non-Dominated Sorting Genetic Algorithm (NSGAII) to improve task scheduling in the cloud [4].

Shukla et al. suggested a multi-objective optimization model that minimizes energy consumption and make span for efficient task scheduling. A Non-dominated Sorting Genetic Algorithm (NSGA) based algorithm to solve the optimization multi-objective task allocation problem was also offered [5].

Panwar et al. introduced a novel TOPSIS-based task scheduling algorithm: TOPSIS–PSO algorithm, which worked on the multi-criteria-based approach where three criteria (execution time, transmission time and cost) were taken to improve the scheduling process [6]. Chakravarthi et al. presented a concurrent workflow scheduling method for heterogeneous distributed environments based on the MCDM method TOPSIS. The optimal resource was selected among the available resources as per the workflow task requirements based on the weighted sum of execution time, cost and communication time [7].

Khorsand et al. proposed an energy-efficient task scheduling algorithm operational on BWM and the TOPSIS in [8]. BWM process is applied to assign the importance weights for each criterion in their work where the evaluation metrics include the makespan, energy consumption, and resource utilization.

Nayak et al. suggested scheduling deadline-based tasks in cloud computing using the TOPSIS technique. The task selection implemented by them is through a decision-maker due to the presence of different parameters of deadline-sensitive tasks such as start time, the number of VMs, execution time (duration) and deadline [9].

Alla et al. introduced a new strategy the MCPTS, to address the priority issue in both users' requests and providers' resources. The priority is adjusted according to four task parameters including length, waiting time, deadline and burst time. Three sub-models: task priority, task queuing priority and resources priority were investigated. ELECTRE III and differential evolution were proposed to evaluate and determine tasks' priorities [10].

However, none of this reviewed literature used EWM to apply to the TOPSIS and evaluated seven performance metrics. In this work, we have identified the application of entropy in resource management to schedule VMs with the help of TOPSIS. The novel EBTASIC algorithm for resource scheduling by the EWM-based TOPSIS method is proposed in this work. This work aims to extract weights from various scheduling parameters and, based on their weights, deem fit resource lease requests are scheduled in consonance with the ranking obtained from the TOPSIS method.

2 Methodology

Haizea is a well-known open-source resource lease manager developed in Python [11]. It is used to provision computing resources in the form of VMs. A new VM simulator system ViMSiS, similar to Haizea is developed in Java to perform the experiments. The ViMSiS source code is freely accessible on [12] to the scientific community. The main aim of ViMSiS is to design and develop various existing and novel scheduling algorithms. The architecture of ViMSiS is shown in Figure 1. Components and working of ViMSiS, types of lease requests, proposed and supporting algorithms, and EBTASIC flow are presented in the following subsections.

2.1 Components and working of ViMSiS

ViMSiS has mainly eight components as shown in Figure 1. The component resource request handler is to handle lease requests by users. These lease requests can be imported from a comma-separated values (CSV) file or they can be input by the user through the lease request interface. A synthetic workload generator is also provided to generate new lease requests randomly in bulk to experiment and check the efficacy of scheduling algorithms. The resource manager component takes care of available resources to schedule. The entropy weighting calculator is to calculate automatic weights of different attributes in lease requests.

Scheduling Decision Analyzer and Maker is the component where all the new and existing scheduling algorithms can be coded. VM manager acts as a hypervisor that takes care of VM allocation, reallocation, VM suspension and VM relinquishment. The accounting billing and metering component is responsible for bookkeeping, and accounting. Finally, the ViMSiS database stores all logs and adequate leases and VMs information.

ViMSiS may operate in three modes: it can produce synthetic workloads, import workloads from external files, or take user input. The experiments are performed in the first mode. Basic attributes' values were generated randomly by the simulator and derived attributes' values were calculated according to standard formulae on their own. Similarly, evaluation of lease execution, lease request acceptance and rejection, and performance are calculated by ViMSiS. The proposed algorithm EBTASIC is adapted

into the Scheduling Decision Analyzer and Maker (SDAM) of ViMSiS.

2.2 Lease requests types

CSUs request their requirement of hardware resources in the form of lease requests. Lease requests can alternatively be referred to as tasks or cloudlets. ViMSiS, like Haizea, primarily allows three types of lease requests. VMs can be offered by CSPs to fulfil these lease requests. CSPs provision different VM configurations based on a tariff. VM configuration may be fixed or may be customized in line with CSU's requirements and CSPs capabilities. Various types of leasing policies are adapted by CSUs. Literature on leasing policies reveals that there are mainly three leasing policies namely Advance Reservation (AR), Best Effort (BE) and Immediate leasing policies [11,12].

AR policy is useful when the exact requirement of resources is known in well advance. So the resources are reserved with the help of AR lease types before the actual point of time of their requirement. An immediate leasing policy is useful when CSU is not aware of the resources required and their quantity in advance. BE leasing policies are useful when the exact amount of resources is known but their acquisition time is not so important (that may be possible in scientific workloads). BE leases can be queued and executed when the system load is low; also BE leases can be executed in the pre-emptive mode of operation [11,12].

Queued BE leases can be scheduled according to various criteria such as shortest job first (or shortest length/makespan first), earliest deadline first, shortest remaining time first, highest cost/ revenue generation first, lowest energy requirement first etc. BE leases can also be scheduled such as trust-based scheduling; regret management enabled trust-based scheduling, security-aware scheduling etc. Taking a single criterion into account for scheduling leads to improvement in only that criterion and may lead to suffering other criteria. MCDM-based scheduling does not guarantee fascinating improvements in all criteria but exhibits a balanced improvement.

In the presented algorithm EBTASIC, many basic attributes of lease requests have been taken into account viz. VCPU, GPU, memory, extended memory, local SSD, pricing, makespan, revenue generation, and a minimum response time that makes it a deadline-based scheduling algorithm i.e. the length of time up to which execution of a particular lease must be started otherwise it will be rejected. Scheduled execution, lease accepted or rejected, and time taken are derived attributes; their values are calculated according to input attributes. Objective weights of all basic attributes are calculated by a well-defined method EWM, and all these weights are different in values with a total sum equal to one.Performance evaluation of EBTASIC is demonstrated through different metrics such as average waiting time, throughput, response time, average turnaround time, total revenue generated etc

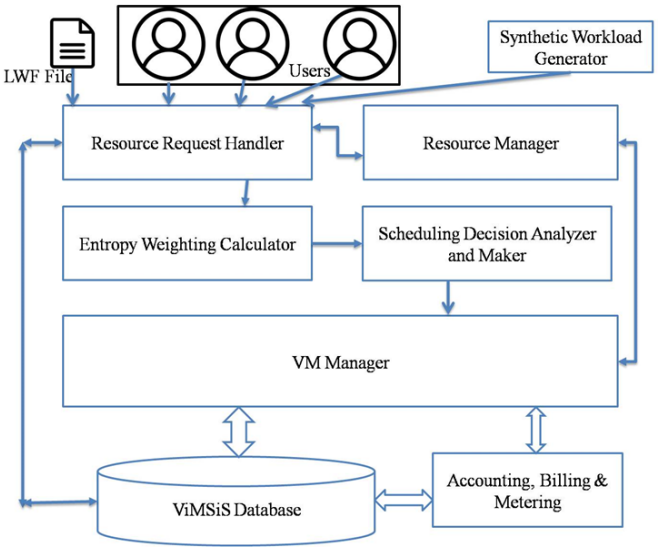

Fig 1. The ViMSiS architecture

2.3 Proposed algorithm

EBTASIC and all other supporting algorithms are shown in this section. Here algorithm 1 is the main EBTASIC algorithm, while Algorithm 2 is to calculate weights of lease request weights, algorithm 3 is adapting the TOPSIS method and algorithm 4

is for the execution of lease requests with the calculation of various performance measures. This subsection also shows the flow of the EBTASIC algorithm in Figure 2.

2.3.1 EBTASIC algorithm

Algorithm 1: EBTASIC Algorithm

1. **Require: X←Lease requests for VMs with VCPU, GPU, Memory, Extended Memory, Local SSD, Minimum Response Time, Price and Makespan**
2. **Procedure: CalculateEBTASIC (X)**
3. **Call Procedure:W=CalculateEntropy (X)**
4. **Call Procedure:M=CalculateRanking (X,W)**
5. **Call Procedure:VMScheduler (M)**
6. **Store values of Average Waiting Time, Average Response Time, Average Turnaround Time, Average Throughput and Total Revenue in the database.**

Algorithm 2: Calculate Lease Request Entropy.

1. **Require: X←Lease requests for VMs with VCPU, GPU, Memory, Extended Memory, Local SSD, Minimum Response Time, Price and Makespan**
2. **Procedure: CalculateEntropy (X)**
3. R_{ij}=$X_{ij}/\sum_{i=1}^{n} X_{ij}$
hereX$_{ij}$=Performance value of ith alternative over jth criteria
4. e_j=-h$\sum_{i=1}^{m} R_{ij}\ln(R_{ij})$, j=1,2,3…,n
5. h=1/ln(m) where "m" is number of alternatives.
6. W_j=1-e_j / $\sum_{j=1}^{n}(1-e_{ij})$
7. Return W

Algorithm 3: Calculate Lease Request Ranking.

1. **Require: X←Lease requests for VMs with VCPU, GPU, Memory, Extended Memory, Local SSD, Minimum Response Time, Price and Makespan**
2. **Require: W← Resource attributes entropy- based weight**
3. **Procedure: CalculateRanking (X,W)**
4. Normalize the lease request vector X by using the formula $\bar{X}_{ij} = \frac{X_{ij}}{\sqrt{\sum_{i=1}^{n} X_{ij}^2}}$
5. Multiply normalized lease request vector by respective weights $V_{ij} = \bar{X}_{ij} \times W_j$
6. Calculate the ideal best and ideal worst value of every column of the lease request matrix. Ideal best V_j^+ is selected in the way that Non-beneficial criteria need minimum value and for beneficial criteria, the maximum value is desired and vice-versa is selected for the ideal worst V_j^- values.
7. Calculate the Euclidean Distance from the ideal best value. $S_i^+ = \sqrt{\sum_{j=1}^{m}\left(V_{ij} - V_j^+\right)^2}$
8. Calculate the Euclidean Distance from the ideal worst value. $S_i^- = \sqrt{\sum_{j=1}^{m}\left(V_{ij} - V_j^-\right)^2}$
9. The performance score of every lease request is calculated as $P_i = \frac{S_i^-}{S_i^+ + S_i^-}$

Assign Ranks to lease requests according to performance score such that the highest performance gets the lowest rank number which is 1.

M←X where requests are arranged according to Ranks.

10. Return W

Algorithm 4: Schedule Lease Requests according to Ranking.

1. **Require: M←Lease requests for VMs with VCPU, GPU, Memory, Extended Memory, Local SSD, Minimum Response Time, Price and Makespan arranged according to rank.**
2. **Procedure: VMScheduler (X)**
3. for i in range (1, No_of_Lease_Requests)
4. if (Resource_Available=='True'&&Execution_Start_Time>= Minimum_Response_Time) then
Schedule VMs according to their Ranks, highest rank first.
else
Reject the lease request and move to the next lease request.

5. Calculate Average Waiting Time, Average Response Time, Average Turnaround Time, Average Throughput and Total Revenue generated.

2.3.2 EBTASIC Flow

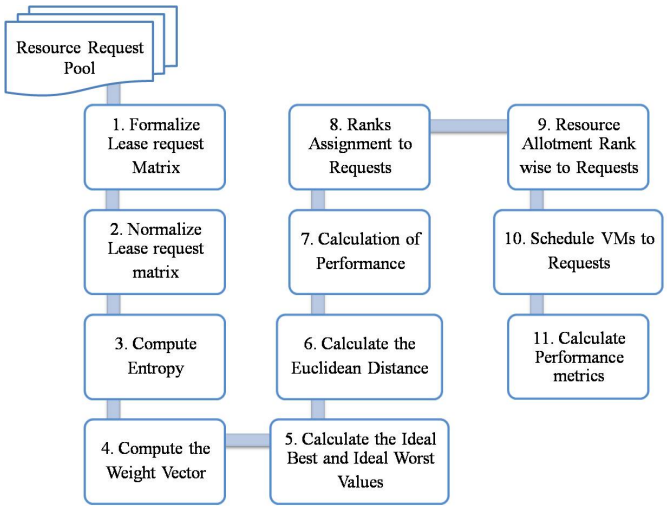

Fig 2. EBTASIC flow

2.3.3 Performance Improvement formula

The following formula is used to compute the performance increase in percent of various performance measures:

$$Improvement\ Percent\ = \frac{New\ value - Old\ value}{Old\ value} * 100$$

In this case, the new value is the performance gained by EBTASIC on a performance measure, whereas the old value is the performance derived from the same metric using the existing FCFS or EDF algorithm.

3 Results and Discussion

To perform the experiments, 12750 deadline-sensitive lease requests were generated by ViMSiS. These lease requests were scheduled using the well-known FCFS, EDF, and proposed EBTASIC algorithms. EBTASIC ensures deadline-based nature and achieves balancing in multi-objective optimization. Figures 3, 4, 5, 6, 7 and 8 compare six parameters for FCFS, EDF, and EBTASIC algorithms: revenue generation, execution time, actual response, turnaround time, throughput, and resource utilization. According to the results, the EBTASIC algorithm outperforms the basic FCFS and EDF algorithms in all seven measures.

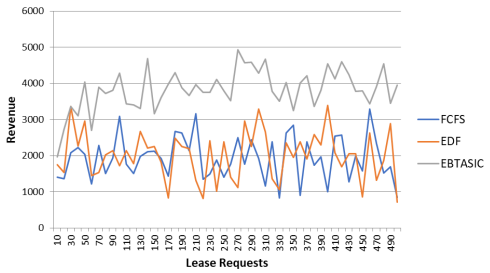

Fig 3. The Revenue generation using FCFS, EDF and EBTASIC algorithms

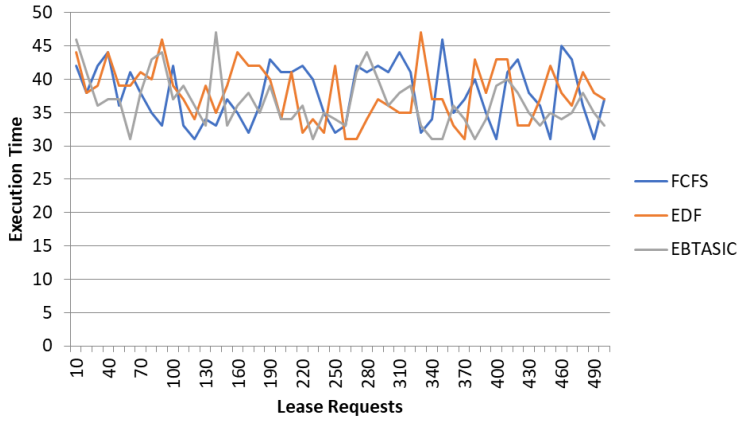

Fig 4. The execution time using FCFS, EDF and EBTASIC algorithms

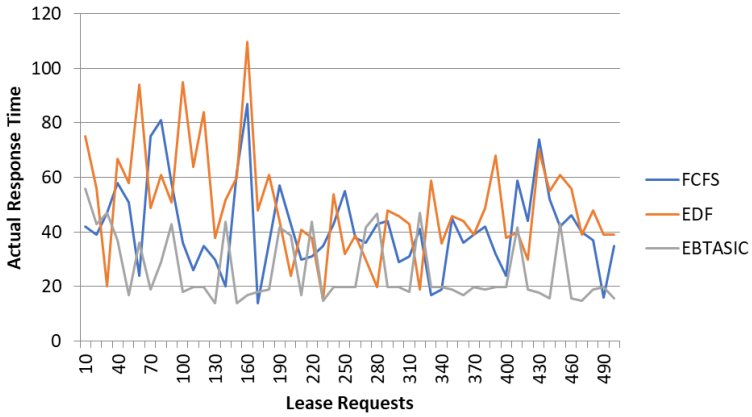

Fig 5. The actual response time using FCFS, EDF and EBTASIC algorithms

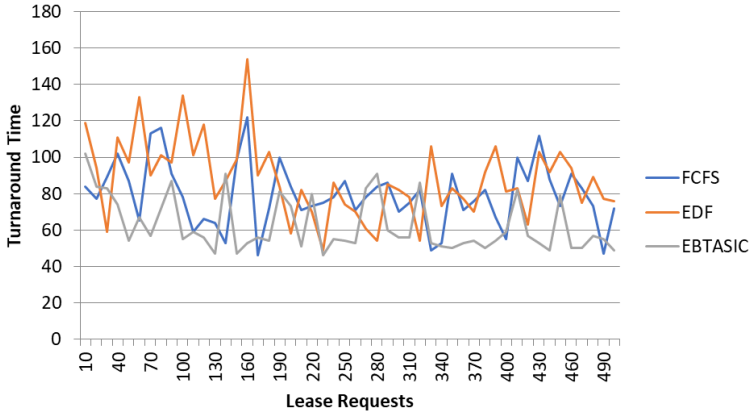

Fig 6. The turnaround time using FCFS, EDF and EBTASIC algorithms

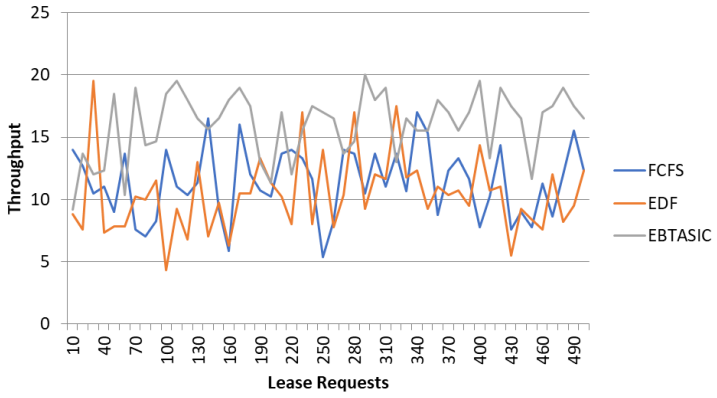

Fig 7. The throughput using FCFS, EDF and EBTASIC algorithms

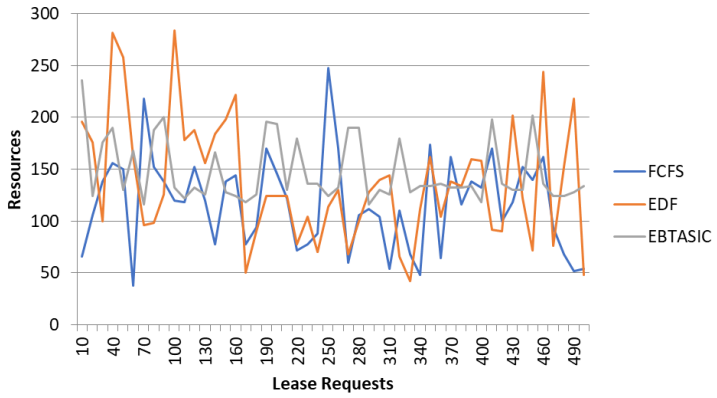

Fig 8. The resource utilization using FCFS, EDF and EBTASIC algorithms

The proposed approach in this study is based on independent lease execution, whereas the work presented in [7], and [13–15] used Entropy-based TOPSIS in different environments or for different problems, making a clear relationship between the performance evaluation of the presented work and the previous work in [7], and [13–15] unfeasible. Subsection 3.1 compares the results of the presented study to prior studies.

Comparative Analysis

MCPTS, a priority task scheduling algorithm, is proposed in [10]. This algorithm adjusts the priority based on parameters such as task length, waiting time, deadline, and burst time. MCPTS is pitted against the EDF and FCFS algorithms. The MCPTS outperforms EDF and FCFS in terms of resource utilisation by 44.93% and 61.47%, respectively. The MCPTS reduces makespan by 20.36% and 58.32% when compared to the EDF and FCFS algorithms, respectively.

Interference Aware Resource Provisioning and Scheduling Approach (IARPS) is suggested in [16]. Comparing the proposed approach to EDF and other cutting-edge methods, resource utilization is improved by 17.26%. It is suggested in [17] to use the fuzzy priority deadline-based task scheduling algorithm (FPDSA). When compared to EDF, FPDSA has an average actual execution time that is 27.94% faster.

In [18] Modified implementation of the Bin Packing Task Scheduling algorithm (MBPTS) is proposed. Particle Swarm Optimization (PSO) and the FCFS algorithm are used to compare the performance of the MBPTS. The simulation results show that the MBPTS reduces average waiting time and makespan by 80.74% and 31.78% respectively, and increases resource utilization by 46.58%.

The experimental results reported in this study show a comparison of the proposed EBTASIC algorithm with the results of well-established FCFS and EDF algorithms. The novelty of the proposed work is the application of entropy in resource

management to schedule VMs with the help of TOPSIS. EBTASIC applies EWM to TOPSIS for resource scheduling. When the experimental results of EBTASIC are compared with the algorithms and experimental results in available literature; it is found that EBTASIC shows improvement in seven performance metrics.

The algorithms proposed in [10–13], and [16–18] also show comparative studies with EDF and/or FCFS but none of these assessed the proposed work on the seven performance measures along with meeting the deadline. Also, none of the proposed works in reviewed literature used EWM to apply to the TOPSIS. Other algorithms proposed in related studies may perform well in some criteria but fail to perform well in others. EBTASIC may perform less well than others in some criteria, but it provides a balanced score across all criteria. Table 1 summarises a comparison of the criteria scope of a few previously proposed algorithms with the proposed algorithm EBTASIC:

Table 1. Comparison between criteria coverage of several previously proposed algorithms

Algorithms	Execution Time	Actual Response Time	Turnaround Time	Throughput	Average Response Time	Resource Utilization	Revenue Generation	Deadline
MCPTS [10]	✔		✔			✔		
IARPS [16]						✔		
FPDSA [17]	✔				✔			
MBPTS [18]	✔		✔		✔	✔		✔
EBTASIC	✔	✔	✔	✔	✔	✔	✔	✔

4 Conclusion

Sequencing lease requests from a wide pool of requests with multi-objective optimization is a tedious task. This work presented a new algorithm EBTASIC based on the well-known MCDM technique TOPSIS with objective weights calculated by the EWM. Beneficial and non-beneficial criteria are given adequate importance while calculating Euclidean distance from ideal best and worst values. It maintains a balance between various criteria with different values and gives a performance score to every lease request. Ranks based on performance scores are used to sort and execute lease requests. Lease acceptance and revenue generation are observed high while resource consumption is observed low using the proposed algorithm. The architecture of the ViMSiS simulator is also presented in this work.

Future work can help in introducing new criteria for the evaluation and ranking of lease requests. Application of the proposed algorithm in container request ranking and scheduling is another direction of future work.

References

1) Cinelli M, Kadziński M, Miebs G, Gonzalez M, Słowiński R. Recommending multiple criteria decision analysis methods with a new taxonomy-based decision support system. *European Journal of Operational Research*. 2022;302(2):633–651. Available from: https://doi.org/10.1016/j.ejor.2022.01.011.
2) Alashaikh A, Alanazi E, Al-Fuqaha A. A Survey on the Use of Preferences for Virtual Machine Placement in Cloud Data Centers. *ACM Computing Surveys*. 2022;54(5):1–39. Available from: https://doi.org/10.1145/3450517.
3) Nayak SC, Parida S, Tripathy C, Pati B, Panigrahi CR. Multicriteria decision-making techniques for avoiding similar task scheduling conflict in cloud computing. *International Journal of Communication Systems*. 2020;33(13):e4126–e4126. Available from: https://doi.org/10.1002/dac.4126.
4) Shariat SS, Barekatain B. HATMOG: an enhanced hybrid task assignment algorithm based on AHP-TOPSIS and multi-objective genetic in cloud computing. *Computing*. 2022;104(5):1123–1154. Available from: https://doi.org/10.1007/s00607-021-01049-y.
5) Shukla DK, Kumar D, Kushwaha DS. Task scheduling to reduce energy consumption and makespan of cloud computing using NSGA-II. *Materials Today: Proceedings*. 2021. Available from: https://doi.org/10.1016/j.matpr.2020.11.556.
6) Panwar N, Negi S, Rauthan MMS, Vaisla KS. TOPSIS–PSO inspired non-preemptive tasks scheduling algorithm in cloud environment. *Cluster Computing*. 2019;22(4):1379–1396. Available from: https://doi.org/10.1007/s10586-019-02915-3.
7) Chakravarthi KK, Shyamala L, Vaidehi V. TOPSIS inspired cost-efficient concurrent workflow scheduling algorithm in cloud. *Journal of King Saud University - Computer and Information Sciences*. 2022;34(6):2359–2369. Available from: https://doi.org/10.1016/j.jksuci.2020.02.006.
8) Khorsand R, Ramezanpour M. An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing. *International Journal of Communication Systems*. 2020;33(9):e4379–e4379. Available from: https://doi.org/10.1002/dac.4379.
9) Nayak SC, Tripathy C. An Improved Task Scheduling Mechanism Using Multi-Criteria Decision Making in Cloud Computing. *International Journal of Information Technology and Web Engineering*. 2019;14(2):92–117. Available from: https://doi.org/10.4018/IJITWE.2019040106.
10) Alla HB, Alla SB, Ezzati A, Touhafi A. A novel multiclass priority algorithm for task scheduling in cloud computing. *The Journal of Supercomputing*. 2021;77(10):11514–11555. Available from: https://doi.org/10.1007/s11227-021-03741-4.
11) Panda SK, Nanda SS, Bhoi SK. A pair-based task scheduling algorithm for cloud computing environment. *Journal of King Saud University - Computer and Information Sciences*. 2022;34(1):1434–1445. Available from: https://doi.org/10.1016/j.jksuci.2018.10.001.
12) Shrivastava V. Virtual Machine Simulator System. ViMSiS 2020. 2021. Available from: https://vimsis.web.app/.

13) Kannan D, Thiyagarajan R. Entropy based <scp>TOPSIS</scp> method for controller selection in software defined networking. *Concurrency and Computation: Practice and Experience*. 2022;34(1):6499–6499. Available from: https://doi.org/10.1002/cpe.6499.
14) Afzalibehbahani N, Khodadadi-Karimvand M, Ahmadi A. Environmental Risk Assessment Using FMEA and Entropy Based on TOPSIS Method: a Case Study Oil Wells Drilling. *Big Data and Computing Visions*. 2022. Available from: https://doi.org/10.22105/bdcv.2022.331778.1054.
15) Sun F, Yu J. Improved energy performance evaluating and ranking approach for office buildings using Simple-normalization, Entropy-based TOPSIS and K-means method. *Energy Reports*. 2021;7:1560–1570. Available from: https://doi.org/10.1016/j.egyr.2021.03.007.
16) Swain CK, Sahu A. Interference Aware Workload Scheduling for Latency Sensitive Tasks in Cloud Environment. *Computing*. 2022;104(4):925–950. Available from: https://doi.org/10.1007/s00607-021-01014-9.
17) Rajan CDS. RETRACTED ARTICLE: Design and implementation of fuzzy priority deadline job scheduling algorithm in heterogeneous grid computing. *Journal of Ambient Intelligence and Humanized Computing*. 2021;12(6):6073–6080. Available from: https://doi.org/10.1007/s12652-020-02171-z.
18) Chraibi A, Alla SB, Ezzati A. An efficient cloudlet scheduling via bin packing in cloud computing. *International Journal of Electrical and Computer Engineering (IJECE)*. 2022;12(3):3226–3226. Available from: https://doi.org/10.11591/ijece.v12i3.pp3226-3237.