Metaheuristic-Driven Optimization for Efficient Resource Allocation in Cloud Environments

Dr.B.Chitradevi
Assistant Professor,
Department of Computer Applications,
SRM Institute of Science and Technology (Deemed to be University),
Tiruchirapalli - 621 105
citrdevi.b@gmail.com

Dr. A. Immaculate Mercy
Assistant Professor (SG), Department of Computer Applications,
Periyar Maniammai Institute of Science and Technology, Deemed to be University, Vallam,
Thanjavur - 613403

Dr.Shanmugam Ramasrinivasan
Assistant Professor,
Arignar Anna Government Arts and Science College,
Karaikal,
Puducherry,
India.
shanmugamrkkl@gmail.com

***Corresponding Author:"Dr.R.Kadher Farook"**
Assistant Professor
Department of Information Technology
Arul Anandar College
Karumathur 625 514
Madurai dt.
kadherfarook@aactni.edu.in

Abstract

Cloud computing offers diverse resource allocation schemes tailored to meet user requirements. Still, ensuring reliability remains a critical challenge in this dynamic environment. This paper contributes an innovative approach by presenting an efficient and dependable resource allocation framework specifically designed for the complexities of cloud environments. By harnessing the power of metaheuristic optimization algorithms, this research endeavors to significantly elevate resource utilization beyond the limitations of traditional techniques. Through comprehensive simulations and extensive evaluations, the results unmistakably demonstrate the pronounced effectiveness of the proposed method. The reductions observed in network overhead, average response time, and memory load validate the robustness and practicality of this heuristic scheme. Hence, this framework not only presents a promising solution but also stands as a well-crafted and forward-thinking strategy, poised to address the evolving demands of stakeholders within the realm of cloud computing.

Keywords: Resource Allocation, Cloud Environment, Metaheuristic, Load Balancing.

1. Introduction

The evolution of the computer industry has brought forth novel computing paradigms such as cloud computing, revolutionizing the way resources are accessed and managed. The capabilities of cloud computing, offering extensive services and virtual resource accessibility via the internet, have made it a popular choice due to its scalability and cost-efficiency. Virtualization technology, a crucial element in modern data centers, plays a pivotal role in infrastructure-based

cloud services. However, the burgeoning demand for client computing has escalated power consumption, particularly in environmentally conscious data centers, posing challenges in terms of energy efficiency and greenhouse gas emissions.

To address the escalating energy consumption in cloud data centers, various energy-saving techniques have been proposed in recent research. Among these, the migration of virtual machines (VMs) has emerged as a key focus area for reducing the demand on physical machines, thereby impacting energy efficiency significantly [1]. Efficiently distributing the workload among physical computers through distributed load balancing is vital for meeting service level agreements (SLAs) in the cloud environment.

Load balancing techniques are indispensable in cloud environments, ensuring optimal resource utilization, faster responses, and minimized task completion times. Studies underscore the adverse effects of uneven resource distribution on cloud data center performance, emphasizing the critical role of load balancing in sustaining cloud operations [2]. However, the continuous hosting of cloud services by data centers leads to heightened power consumption, elevating operational costs and environmental impact.

Addressing the Virtual Machine (VM) allocation challenge, metaheuristic methods such as the Whale Optimization Algorithm (WOA), Grey Wolf Optimization (GWO), Differential Evolution (DE), and Firefly Algorithm (FA) have been employed to optimize resource usage and curtail power consumption. In this paper, we introduce the FA-KNN (Firefly- K Nearest Neighbour Algorithm) by leveraging the Firefly Algorithm as the foundation. Renowned for its adaptability and simplicity, the FA method exhibits high convergence rates, making it well-suited for VM placement. Additionally, our proposed approach considers a diverse range of resources beyond the central processing unit, aiming to maximize elements like energy efficiency, power consumption, and overall resource utilization.

2. Literature Review

Ali et al. [3] introduced an innovative approach aiming to improve Virtual Machine (VM) allocation, optimize resource utilization, balance multifaceted resources, and reduce communication traffic. Their proposal involved an enhanced chaotic binary Grey Wolf Optimization (GWO) strategy, which effectively addressed the allocation challenges within cloud data centers. In a similar vein, Sasan et al. [4] proposed leveraging a chaotic hybrid optimization algorithm to systematically resolve assignment issues in heterogeneous server environments. Their method streamlined VM placement across diverse servers in cloud data centers, foreseeing and subsequently mitigating energy consumption concerns prevalent in cloud computing infrastructures. These pioneering approaches emphasize the significance of optimized resource allocation and energy efficiency in the dynamic landscape of cloud-based computing systems.

Adhikari et al. [5] proposed LB-RC aiming to reduce execution costs by utilizing Quality of Service (QoS) options. However, this approach does not consider the qualities of task deployment policies, potentially limiting its effectiveness in optimizing task allocation and resource utilization. Jena et al. [6] introduced QMPSO, a method that seeks to equalize workload

by reassigning loads to appropriate Virtual Machines (VMs) based on each VM's fitness score. Nevertheless, QMPSO is constrained to handling independent tasks, which might limit its applicability in scenarios with interdependent tasks.

Golchi et al. [7] proposed a hybrid approach employing Firefly and Improved Particle Swarm Optimization (IPSO) for task allocation. However, their method resulted in increased task response times, indicating a challenge in maintaining energy efficiency while enhancing load balancing. Haidri et al. [8] introduced CPDALB, emphasizing improved load balancing in heterogeneous environments. Their approach focuses on better load distribution but suggests the need to consider various other parameters apart from load balancing for comprehensive optimization.

Pourghaffari et al. [9] proposed EDF-VD, highlighting its potential to enhance load balancing outcomes, particularly by improving the handling of split tasks. However, there is room for refining task scheduling to achieve more favorable outcomes. Kaur et al. [10] introduced TSFPA, which effectively reduced task makespan compared to other techniques. Nonetheless, the study suggests the necessity of considering task load balance as a critical aspect for further improvement.

Mishra et al. [11] presented ACO-Fuzzy, aiming to reduce costs and optimize computer network routes for resource distribution. They emphasize the need for multiobjective optimization encompassing resource, energy, and task migration for comprehensive improvements. Xiaolong et al. [12] introduced MTSS, projecting better network utilization and reduced packet loss as outcomes. However, their method might overlook varying load circumstances, possibly affecting its adaptability in dynamic environments.

The diverse array of methodologies presented in cloud-based computing systems strive to optimize resource allocation, load balancing, and energy efficiency. While innovative strategies like enhanced optimization algorithms demonstrate promise in addressing VM allocation and energy consumption concerns, some approaches exhibit limitations in handling task deployment policies and considering multiple parameters beyond load balancing. Ongoing research emphasizes the quest for comprehensive optimization in the dynamic landscape of cloud computing systems.

3. Methodology

In the realm of cloud computing architectures, achieving optimized energy-efficient resource management stands as a critical objective. Load balancing techniques play a pivotal role in distributing workloads effectively among servers, aiming to minimize energy consumption while ensuring efficient job scheduling within the cloud environment. The primary aim of such systems revolves around not only enhancing load balancing but also reducing energy consumption by orchestrating workload transfers from overloaded servers to those operating below capacity. Despite numerous systems designed with these objectives, significant reductions in energy consumption can be further attained through a fusion of diverse algorithms. In this context, proposed system models, simulated and replicated through tools like Cloud Analyst, aim to emulate cloud infrastructures. These models adhere to a prescribed methodology that emphasizes

workload balancing and resource allocation strategies, ultimately geared towards the overarching goal of minimizing energy usage within cloud environments.

The newly devised energy-efficient load balancing system delineates a structured process flow that embodies the principles of efficient workload balancing and resource allocation to curtail energy consumption. This system operates through a sequence of steps meticulously designed to optimize the distribution of workloads among servers. By intelligently transferring tasks from overloaded servers to underutilized ones, it endeavors to ensure a balanced workload distribution, thereby mitigating energy wastage. The systematic approach embedded within this load balancing system harnesses algorithmic blends, enabling it to make informed decisions regarding workload allocation while striving to achieve the overarching aim of reducing energy consumption within the cloud infrastructure,

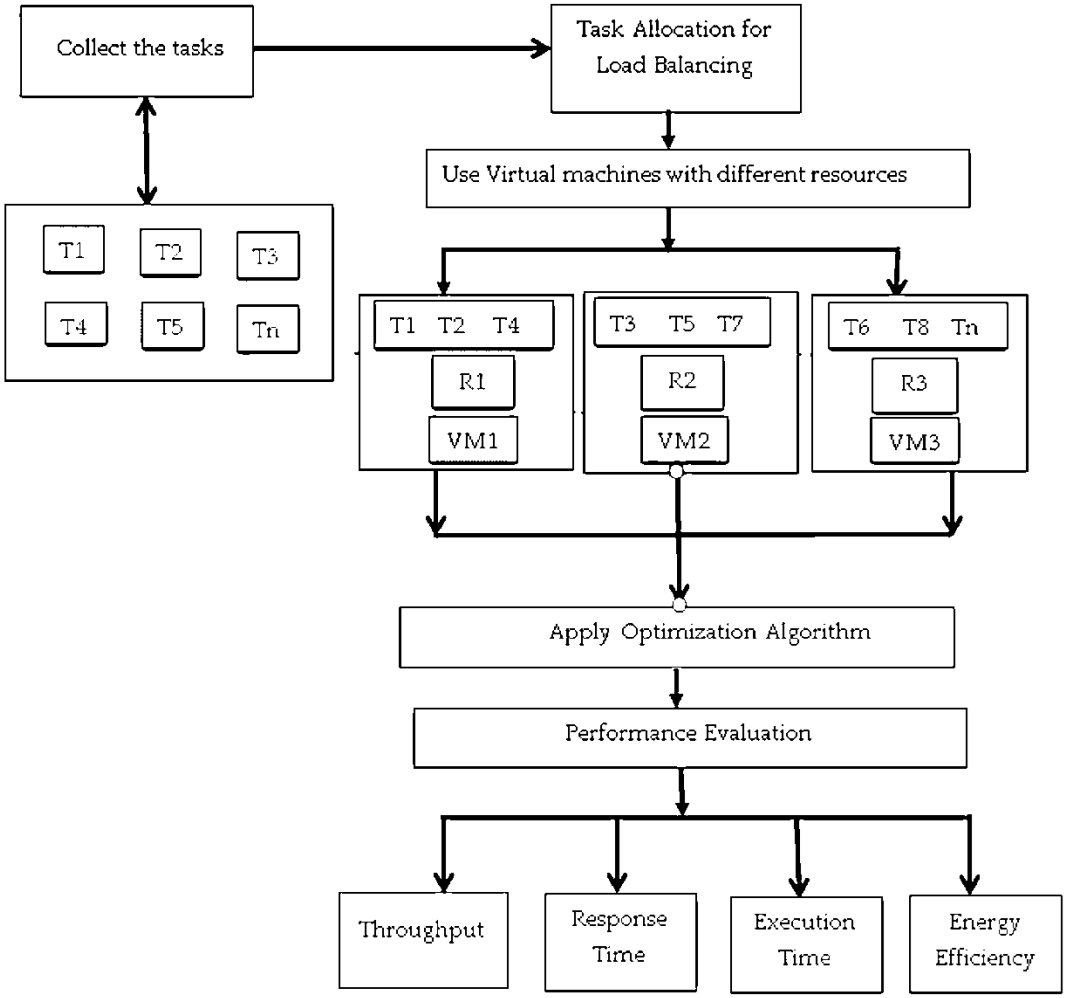

Figure 1. Flowchart of the proposed system

The workflow depicted in the provided figure initiates with the aggregation of multiple jobs and their respective inputs into tasks. These tasks are subsequently allocated to different virtual machines based on the load assessment conducted by a scheduler. A proposed technique assesses the performance of individual machines concerning their designated functionalities and

undertakes resource redistribution to enhance overall system performance. The recommended approach for achieving optimal energy-efficient load balancing adheres to a specific set of computational procedures. The steps within this methodology are as follows:

- To generate the desired number of tasks, enter the necessary number of parameters.
- To distribute the various resources across each virtual machine, host the specified number of servers.
- Launch all of your virtual machines there and allot storage for them using the data center.
- Use optimization techniques to determine the most effective resource scheduling, load balancing between servers, resource migration, and energy consumption calculations.
- Calculating the energy usage, execution time, and pre- and post-migration times is necessary for cloud load balancing.

4.1 Optimization Algorithms Used for Load Balancing and Energy Efficiency

This section delineates the structured design of the proposed optimized energy-efficient load balancing model, segmented into distinct phases, each employing specialized algorithms to achieve load balancing and resource scheduling within the system. The initial phase involves the implementation of optimization algorithms aimed at streamlining system performance. Subsequently, in the second phase, a combination of optimization and Machine Learning (ML) algorithms is employed to further enhance resource utilization efficiencies. The final phase integrates the application of the whale optimization algorithm, recognized for delivering optimal results in optimizing cloud systems. This section thoroughly explores the utilization and significance of these algorithms within the proposed system.

i. WOA-based method

The Whale Optimization Algorithm (WOA), inspired by humpback whales' hunting behavior, employs three operators resembling the hunting strategies of these marine mammals: searching for prey, circling it, and utilizing bubble nets. Widely utilized in solving optimization problems, the WOA has found applications in various domains. A hybrid version coupling WOA with simulated annealing is proposed for feature selection in (14), aiming to enhance exploration by identifying promising regions revealed by the WOA algorithm. Another study (15) presents a feature selection strategy based on WOA, utilizing crossover, mutation operators, and selection strategies like tournament and roulette wheel to refine the search process within swarm intelligence systems. Nematzadeh et al. (16) suggest a frequency-based filter feature selection method employing the whale algorithm, effectively excluding insignificant features. Additionally, this strategy employs Mutual Congestion as a distinct filtering technique to evaluate remaining features in highly dimensional medical datasets.

ii. GWO-based method

The Gray Wolf Optimization (GWO) algorithm, inspired by the hunting behaviors of grey wolves, embodies a novel meta-heuristic approach. It simulates the hierarchical leadership structure and hunting strategies observed in nature among grey wolf packs. In recent years, GWO-based methodologies have found application in data mining, particularly in feature selection. Utilizing a Multi-Objective GWO, researchers have successfully identified pertinent and non-redundant features, augmenting wrapper model performance through a hybrid approach that integrates the simplicity of the filter model's computational structure. Additionally, a binary GWO variant has been proposed specifically for selecting optimal feature subsets in classification tasks. This variant's fitness function in the wrapper-based feature selection technique considers both classification accuracy and the number of chosen features. To further enhance GWO-based methods, a multi-strategy ensemble GWO has been introduced for feature selection, aiming to improve the selection process. Abdel-Basset et al. (Abdel-Basset, El-Shahat et al.) introduced a GWO-based wrapper feature selection technique that incorporates a Mutation operator for data classification in their 2020 paper. This operator focuses on removing redundant features and introducing new features to enhance classification accuracy, thereby refining the prior GWO-based approach.

iii. Differential Evolution (DE) method

The Differential Evolution (DE) algorithm, a swarm intelligence-based evolutionary approach, has emerged as a robust tool in addressing diverse optimization challenges. Unlike genetic algorithms, DE emphasizes local search, rectifying a primary deficiency of genetic selection operators. Notably, its applications in pattern recognition and feature selection have gained attention. Al-Ani et al. introduced a wrapper-based feature subset selection method that combines differential evolution with a wheel-based search technique (17). Their approach efficiently removes redundant features, presenting a multi-objective differential evolution-based strategy to identify the most relevant feature sets. Another advancement, the Binary Differential Evolution with Self-Learning (MOFS-BDE), is proposed in (18), employing a new mutation operator to navigate away from local optima. Additionally, a novel multi-objective differential evolution technique for feature selection is suggested in (19), aiming to simultaneously enhance the performance of clustering algorithms. These iterations showcase the adaptability and efficacy of the DE algorithm in diverse optimization scenarios.

iv. FA-based methods

The Firefly Algorithm (FA) draws inspiration from the communication between fireflies, reflecting swarm intelligence and enabling cooperation between low and high-performing agents to achieve optimal results. Recent advancements in FA-based feature selection methods, like the return-cost based binary FFA, aim to enhance feature selection accuracy by addressing the algorithm's premature convergence and refining the selection precision. These novel approaches infuse traditional FA with modifications that not only improve feature selection precision but also prevent local optimization traps and expedite convergence. Additionally, a feature selection strategy leveraging FA is proposed (20), employing an SVM classifier to categorize Arabic texts post the final feature selection. Furthermore, an FA-based feature selection method for network intrusion detection optimizes the feature set using both wrapper-based (C4.5 and Bayesian network) and filter-based (Mutual Information) selection methods, ensuring a refined final

feature selection process. These advancements highlight the adaptability and versatility of FA in enhancing feature selection across diverse domains.

v. K-Nearest Neighbour (KNN) Algorithm

K-Nearest Neighbor is a supervised learning algorithm where the result of new instance query is classified according to scale of K nearest neighbor categories. The key to the Knearest neighbor algorithm is to identify those K nearest neighbors in the training set. The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the classification phase, K is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the K training samples nearest to that query point. The degree of proximity between query instance and K nearest neighbors is determined by measuring the Euclidean distance formula, between them. According to Euclidean distance the Euclidean distance between points a = (al'a2,"',a,,) and b = (b"b2, .. ·,bJ is given by:

$$d(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

When it comes to processing massive high-dimensional data sets, one shortcoming of the traditional K-nearest neighbor algorithm is the time complexity of making classification. Since the traditional K-nearest neighbor algorithm is a lazy learning method. It needs to store all the training samples before the classification. A query instance should calculate its distance to all training samples, which may lead to considerable overhead when the training data set is large. Additionally, the algorithm is based on the assumption of that all the classes of the samples have the same percentage in the training set. It is obviously not always consistent with actual situation. As class population is unbalanced, for example one class has large sample size while others have small size in the training set, it would cause erroneous judgment [21].

4.2 Proposed system

Suppose that there are m tasks T={t_1,t_2,,...,t_m}, and n virtual machines VM={⟦vm⟧_1,⟦vm⟧_2,...,⟦vm⟧_n}. It is a NP-complete problem which has n^m ways to allocate these tasks to VMs. When a task is bound to a virtual machine ⟦vm⟧_i the occupation time of ⟦vm⟧_j depends on the length of task and the CU of the VM [14]. The execution time for t_ij on vmj is given by t_ij

t_ij=L_i/C_j

where i∈{1,2,…,m},
j∈{1,2,…,n},
L_(i) represents the length of task t_i,
and C_j denotes the CU (Compute Unit) of virtual machine ⟦vm⟧_j.

Proposed Method: The population is evaluated and selected to create a new generation by the fitness function. To find the correctness of the schedule the fitness function is used:

fitness (P)=min(U_1,U_2,…..,U_n)

where U_1,U_2,…..,U_n are the chromosomes that represent the time taken to finish all the cloudlets execution for their respective assignment [22] .

Algorithm

- Step 1: Start
- Step 2: Calculate the process priority as per their time
- Step 3: Initialize the population as per the process priority
- Step 4: Evaluate the fitness function to determine the fitness of each individual.
- Step 5: Select the fittest chromosome.
- Step 6: Perform crossover mapping over chromosomes by applying KNN algorithm.
- Step 7: Perform mutation by changing the genes of individual parents.
- Step 8: Add the chromosome to a new population
- Step 9: Repeat steps 2 to 7 for all new arrival process
- Step 10: Exit

Cloudlet: In CloudSim, a Cloudlet represents the workload slated for execution during the simulation run within the CloudSim simulation engine. It stands as a significant model within the 'org.cloudbus.cloudsim' package, delineating the specifications for the simulation engine, mirroring real-world application scenarios that could potentially transition to a Cloud-based infrastructure. Essentially, a Cloudlet can be described as a singular task or process enacted within a Cloud-based system, serving as the subject for simulation within the CloudSim simulation engine [22].

Cloud Broker or Datacenter Broker: The Cloud Broker or Datacenter Broker is a model class designed to facilitate negotiations between Software as a Service (SaaS) entities and Cloud providers. These negotiations primarily revolve around Quality of Service (QoS) requirements. Acting as a mediator for applications, this broker class interfaces with the Cloud Information System (CIS) to identify appropriate resources or services. Additionally, it engages in negotiations to secure the allocation of resources or services aligned with the specific QoS demands of the application. Furthermore, this class serves as a foundation for the assessment and experimentation of tailored brokering policies [22].

Steps for Resource Scheduling:

1. Create a container to store the VMs which can be passed to a broker later.
2. Pass VM Parameters.
3. Create a container to store the cloudlet.
4. Pass cloudlet parameters.
5. Create Data Centers.
6. Create Broker.
7. Create an Initial Population.
8. Calculate the fitness index of the chromosome that is calculated by the time required for the completion of each task.

9. Perform Crossover between the different chromosomes gene's lists with KNN algorithm.
10. Perform Mutations in the gene's list.
11. Calculate the fitness function and replace the old population by new population
12. Repeat Step 9-11 until the stopping criteria condition is met.
13. Take the chromosome that has the highest fitness index.
14. Separate the final VM list and cloudlet list from the chromosome.
15. Pass the list to the broker for simulation.
16. Simulation will be finished after every cloudlet are assigned to VMs and finished their execution.

4. Result and Discussion

This section focuses on the computational tests that are performed to determine how well the Tproposed method performs. The Cloudsim tool was used to simulate the proposed approach. The fundamental substrate of this toolkit is Java. All of these tests were performed on a machine equipped with an Intel(R) Core(TM) i5-457 processor, four 2.9 GHz Processors, eight GB of RAM, and a 64-bit Windows operating system. In this section, we examine the simulation outcomes that required the least amount of time and energy. Table 2 shows how the parameters are represented. This experiment employs two distinct datasets to assess the efficacy of the proposed approach. They have 100-500 tasks in each distribution, which is biased to the left and right. Unlike right skewed, which has more little sized activities and fewer large sized obligations, left skewed has less small sized jobs and more large sized tasks [22].

Simulation: Cloud computing is required to see the application of real-time outlines, which calls for simulation. It can be used with a variety of tools, including Cloudsim, CloudAnalyst, iCanCloud, GroundSim, NetworkSim, SPECI, and Cloud Report. In this study, we make use of Cloudsim and CloudAnalyst.

CloudSim: A program called Cloudsim is used to model, simulate, and run tests on cloud computing infrastructures and services. The Cloud Computing and Distributed Systems Laboratory at the University of Melbourne developed the tool. It is being used in this research to implement various resource scheduling algorithms, including the First Come First Serve, Round Robin, Shortest Resource First, and Primarily Genetic algorithms. It may create several data centres with actual host machines and storage servers inside. These devices house several virtual machines running various cloudlets. The simulation of allocating and running a workload on a cloud infrastructure is possible with CloudSim [22].

Table 1. List of Parameters

Parameters	**Value**
Number of data center	5
Number of host	10
Host memory capacity	10 GB
Host bandwidth	2800 Mbps
Number of VMs	50
VM policy	Time_shared

VMM	Xen
Number of vCPU	[1-5]
Task MIPS	[200-15000]

5.2 Performance Analysis

In this section, a comparative analysis between the proposed algorithm and several existing algorithms, including WOA, GWO, DE, and FA, regarding MakeSpan, Energy consumption, and Resource utilization is presented. The table below illustrates a comprehensive comparison of these metrics observed from the implementation of these algorithms across various research phases.

Table 2. Comparative analysis of proposed algorithm

Algorithms	MakeSpan	Energy consumption	Resource utilization
WOA	67	96	58
GWO	66	94	59
DE	63	94	60
FA	62	93	61
FA-KNN	61	89	63

i. MakeSpan

Makespan refers to the longest possible time that a cloud system will take to execute a task. The time Eji required by the resource Rj and PEj to perform the task Ti represents the overall time required by the resource Rj to complete all the tasks. Makespan is determined by the equation below [22].

$$F(E) = \max\left\{V_{j=1}^{n} E_j\right\}$$

Where E_j represents job j's completion. The scheduling algorithm is more effective the shorter the makespan. The suggested system's primary objective is to reduce makespan value. Here, we contrast a number of currently used algorithms, including PSO, Whale, GA, and our suggested approach, to assess the effectiveness of the makespan value.

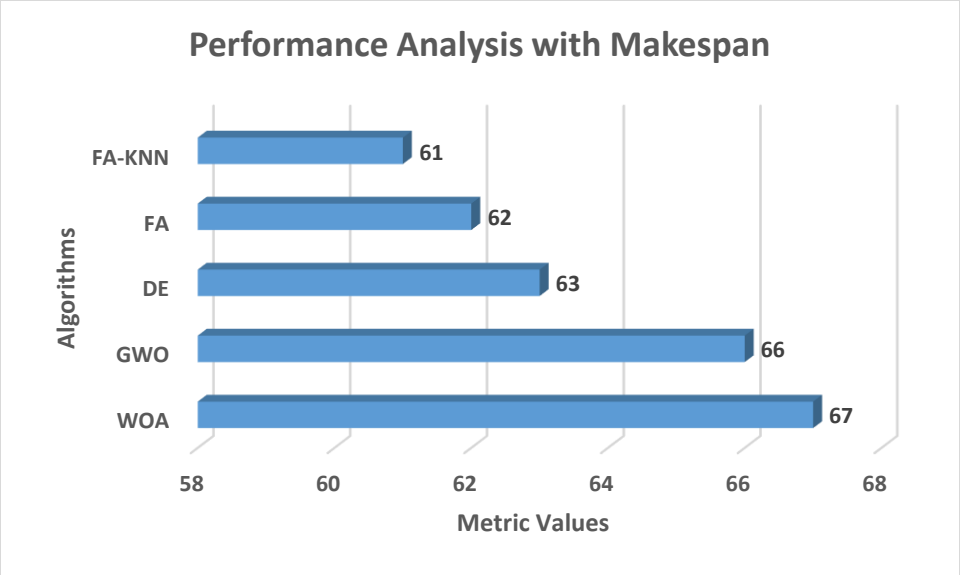

Figure 2. Comparative analysis with Makespan

The figure above calculates the makespan values across different VM counts (100, 200, 300, and 400) by contrasting existing algorithms with the proposed algorithm. Notably, our proposed method demonstrates minimal makespan values in comparison to the existing methods. These findings suggest that the results obtained from our proposed method exhibit a significant improvement, particularly in the makespan metric, when compared against other algorithms. Hence, the outcomes presented by the proposed method underscore its enhanced performance and its potential for broader usage.

ii. Energy consumption

The following figure illustrates a graph comparing energy consumption as the task count varies from 100, 200, 300 to 400. Notably, the proposed algorithm exhibits lower energy consumption compared to other existing algorithms. This trend demonstrates the efficiency of the proposed approach in minimizing energy usage, showcasing its superiority over alternative methods [22].

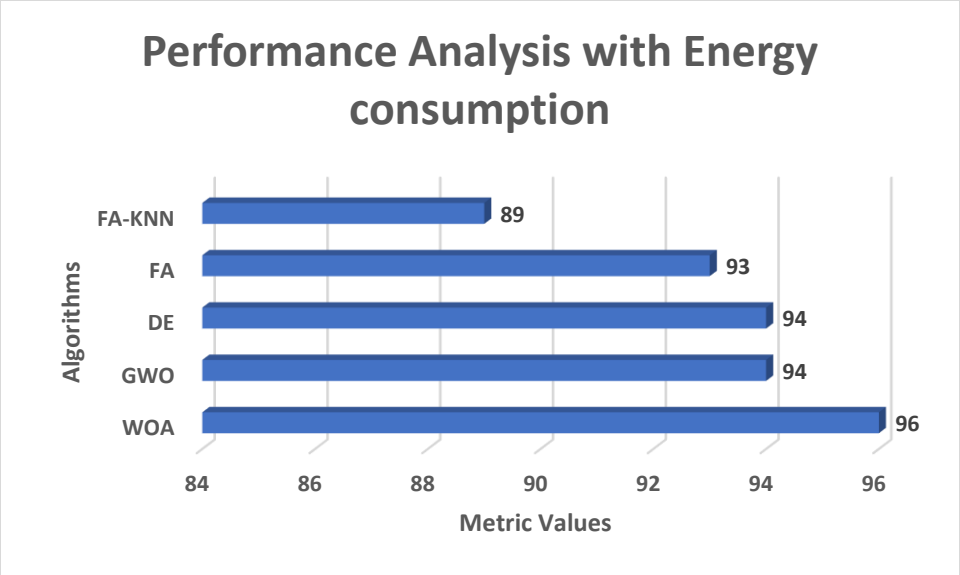

Figure 3. Performance Analysis with Energy consumption

iii. Resource utilization

The figure below displays the average resource utilization of the PSO, Whale, GA, and the Proposed algorithms. Resource utilization plays a critical role in the scheduling task process, significantly benefiting cloud providers. The primary aim is to maximize resource utilization, ensuring optimal usage of resources and maximizing profitability by keeping resources highly occupied. The results indicate that our proposed algorithm consistently maintains a high level of resource utilization, outperforming the other two algorithms. This observation underscores the superior resource management capability of our proposed approach [22].

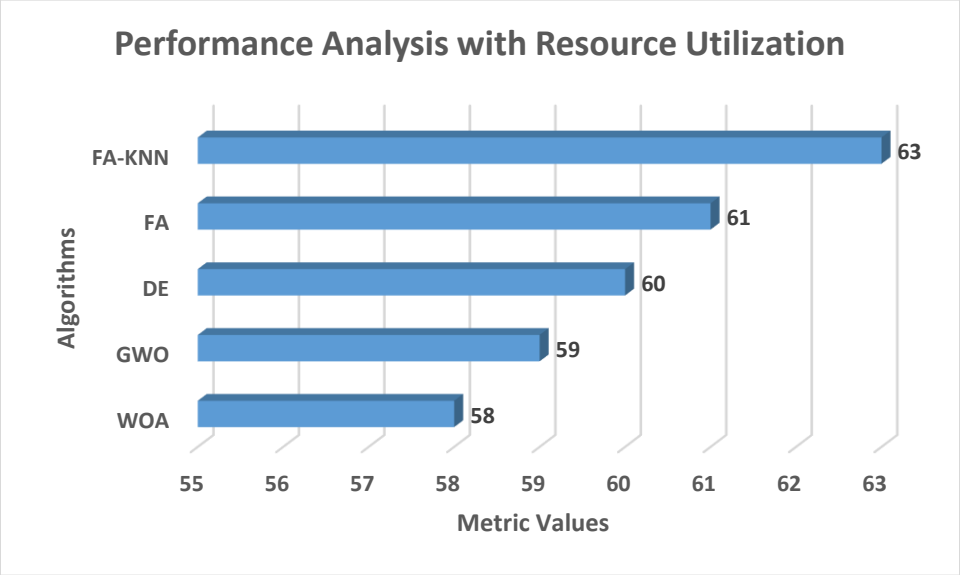

Figure 3. Performance Analysis with Resource Utilization

5. Conclusion

Among the rising demand for cloud services, the pressing challenge of swiftly provisioning resources in cloud environments remains significant. This study introduces an efficient approach utilizing the Firefly Algorithm (FA), an online heuristic resource scheduling algorithm, supported by extensive experimental assessments. The objective was to achieve a balanced workload by strategically allocating Virtual Machines (VMs) based on processing capacities and organizing cloudlets by their lengths. A comprehensive list of VMs and cloudlets undergoes resource allocation via the FA Algorithm facilitated by the broker. The paper conducts a comparative analysis of resource scheduling algorithms, particularly focusing on the performance of the Whale Optimization Algorithm. The FA-KNN algorithm demonstrated remarkable outcomes, including load balancing, enhanced processor utilization, reduced make span and costs, and optimized throughput. Notably, the study showcased the FA algorithm's superior efficacy compared to batch heuristic algorithms, highlighting its performance and effectiveness.

References:

1. P. Kumar, A. Tharad, U. Mukhammadjonov and S. Rawat, "Analysis on Resource Allocation for parallel processing and Scheduling in Cloud Computing," IEEE, 5th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2021, pp. 1-6.

2. D. K. Jain, S. K. S. Tyagi, S. Neelakandan, M. Prakash and L. Natrayan, "Metaheuristic Optimization-Based Resource Allocation Technique for Cybertwin-Driven 6G on IoE Environment," in IEEE Transactions on Industrial Informatics, vol. 18, no. 7, pp. 4884-4892, July 2022.

3. Ali, M.; Masdari, M.; Gharehchopogh, F.S.; Jafarian, A. Improved chaotic binary grey wolf optimization algorithm for workflow scheduling in green cloud computing. Evol. Intell. 2021, 14, 1997–2025.

4. Sasan, G.; Masdari, M.; Jafarian, A. Power efficient virtual machine placement in cloud data centers with a discrete and chaotic hybrid optimization algorithm. Clust. Comput. 2021, 24, 1293–1315.

5. Adhikari, M.; Amgoth, T. An Enhanced Dynamic Load Balancing Mechanism for Task Deployment in IaaS Cloud. In Proceedings of the 2018 International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, India, 28–29 September 2018; pp. 451–456.

6. Jena, U.; Das, P.; Kabat, M. Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. J. King Saud Univ. Comput. Inf. Sci. 2020.

7. Golchi, M.M.; Saraeian, S.; Heydari, M. A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation. Comput. Netw. 2019, 162, 106860.

8. Haidri, R.A.; Katti, C.P.; Saxena, P.C. Capacity based deadline aware dynamic load balancing (CPDALB) model in cloud computing environment. Int. J. Comput. Appl. 2019, 1–15.

9. Pourghaffari, A.; Barari, M.; Kashi, S.S. An efficient method for allocating resources in a cloud computing environment with a load balancing approach. Concurr. Comput. Pract. Exp. 2019, 31, e5285.

10. Kaur, J.; Sidhu, B.K. A New Flower Pollination Based Task Scheduling Algorithm in Cloud Environment. In Proceedings of the 2017 4th International Conference on Signal Processing, Computing and Control (ISPCC), Solan, India, 21–23 September 2017; pp. 457–462.

11. Mishra, S.; Sahoo, M.N.; Sangaiah, A.K.; Bakshi, S. Nature-inspired cost optimisation for enterprise cloud systems using joint allocation of resources. Enterp. Inf. Syst. 2021, 15, 174–196.

12. Xiaolong, X.U.; Yun, C.H.E.N.; Liuyun, H.U.; Anup, K.U.M.A.R. MTSS: Multi-path traffic scheduling mechanism based on SDN. J. Syst. Eng. Electron. 2019, 30, 974–984.

13. Xu, M.; Li, G.; Yang, W.; Tian, W. FlexCloud: A Flexible and Extendible Simulator for Performance Evaluation of Virtual Machine Allocation. In Proceedings of the 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), Chengdu, China, 19–21 December 2015; pp. 649–655.

14. Mafarja, M. M. and S. Mirjalili (2017). "Hybrid Whale Optimization Algorithm with simulated annealing for feature selection." Neurocomputing 260: 302-312.

15. Mafarja, M. and S. Mirjalili (2018). "Whale optimization approaches for wrapper feature selection."

16. Nematzadeh, H., R. Enayatifar, M. Mahmud and E. Akbari (2019). "Frequency based feature selection method using whale algorithm." Genomics. Applied Soft Computing 62: 441-453.

17. Abdel-Basset, M., D. El-Shahat, I. El-henawy, Victor Hugo C. de Albuquerque and S. Mirjalili (2020). "A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection." Expert Systems with Applications 139: 112824.

18. Al-Ani, A., A. Alsukker and R. N. Khushaba (2013). "Feature subset selection using differential evolution and a wheel based search strategy." Swarm and Evolutionary Computation 9: 15-26.

19. Zhang, Y., D. Gong and J. Cheng (2017). "Multi-Objective Particle Swarm Optimization Approach for Cost-Based Feature Selection in Classification." IEEE/ACM Transactions on Computational Biology and Bioinformatics 14(1): 64-75.

20. Larabi Marie-Sainte, S. and N. Alalyani (2020). "Firefly Algorithm based Feature Selection for Arabic Text Classification." Journal of King Saud University - Computer and Information Sciences 32(3): 320.

21. D. Yan, X. Yang and L. Cuthbert, "Regression-based K nearest neighbours for resource allocation in network slicing," 2022 Wireless Telecommunications Symposium (WTS), Pomona, CA, USA, 2022, pp. 1-6, doi: 10.1109/WTS53620.2022.9768174.

22. R.KALAIVANI, V. SUJATHA, Energy Efficient and Load Balanced Optimal Resource Allocation framework for Cloud Environment using ML based Meta heuristic techniques, International Journal of Mechanical Engineering, ISSN: 0974-5823, Vol. 7 No. Kalahari Journals, 2 February, 2022.