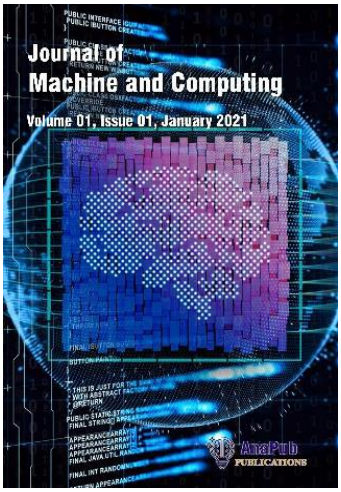
Journal Pre-proof

Advance Hybrid Model in Cloud Computing for Task Scheduling and Resources Allocation Using Meta-Heuristic Machine Learning Model



Manikandan Nanjappan, Chin-Shiuh Shieh and Mong-Fong Horng

DOI: 10.53759/7669/jmc202505128

Reference: JMC202505128

Journal: Journal of Machine and Computing.

Received 29 January 2025

Revised form 12 April 2025

Accepted 13 June 2025

Please cite this article as: Manikandan Nanjappan, Chin-Shiuh Shieh and Mong-Fong Horng, "Advance Hybrid Model in Cloud Computing for Task Scheduling and Resources Allocation Using Meta-Heuristic Machine Learning Model", Journal of Machine and Computing. (2025). Doi: https://doi.org/10.53759/7669/jmc202505128.

AnaPub PUBLICATIONS

Advance hybrid model in cloud computing for task scheduling and resources allocation using meta-heuristic machine learning model

[1]Manikandan Nanjappan*, [2]Chin-Shiuh Shieh, [3]Mong-Fong Horng

[1]Department of Data Science and Business Systems, SRM Institute of Science and Technology, Chennai, Tamil Nadu, India

[1,2,3]Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Taiwan

macs2005ciet@gmail.com, csshieh@nkust.edu.tw, mfhorng@nkust.edu.tw

*Corresponding Author: Manikandan Nanjappan

Abstract

Modern technology requires cloud computing. Allocating resources and scheduling tasks are crucial components of cloud computing. Nondeterministic polynomial completeness (NP) of cloud systems makes job scheduling one of the most challenging aspects of cloud communications. This research proposes novel technique in advancement in hybrid model for task scheduling and resource allocation using meta-heuristic machine learning model in cloud computing networks. Here the cloud network is deployed with numbers of users and clients with virtual machines. The task scheduling model for this deployed network is carried out using convolutional transfer graph proximal policy-based firefly harmony search cat optimization. Then the resource allocation is carried out using software defined virtual machine-based reinforcement markov model. the experimental analysis is carried out in terms of resource utilization, network efficiency, throughput, latency, QoS. For a particular collection of jobs, our primary contribution is to decrease processing time as well as boost speed and efficiency. The proposed technique attained resource utilization of 45%, network efficiency of 96%, throughput of 97%, latency of 95%, QoS of 98%.

Keywords: task scheduling, resource allocation, meta-heuristics, machine learning model, cloud computing networks

1. Introduction

One of the most prevalent paradigms for large-scale data systems is cloud computing. Because it provides a vast quantity of storage as well as resources to many businesses and organisations, which can access these resources with appropriate administration, regulation, and security, cloud computing has become a highly regarded technology worldwide [1]. Given client requirements, many cloud applications need the short-term enhancement of processing power. Increasing available resources would seem to be a straightforward solution to this problem but is not a practical one because of prohibitive costs. Other suggestions are to improve job scheduling algorithms for maximised resource usage, perform online and offline tasks for optimal resource usage, and apply load balancing algorithms to increase utilisation rate. Task scheduling is defined as practice of arranging incoming requests (tasks) in a certain order to accommodate maximum possible usage from the available resources. The customers of service have to submit their requests on the web because cloud computing is essentially the technology for providing services over the Internet. Large queues of customers can generate many requests (or tasks) concurrently. Contrarily, wait for jobs may be very high in systems that do not have any form of scheduling. Additionally, with waiting, short jobs could actually die [2].

The scheduler shall have to consider certain constraints in scheduling process such as nature and size of the job, time for execution, resources available, task queuing, and resource

loading. Scheduling tasks are one of the key problems in cloud computing. Resource allocation as well as task scheduling are thus the two sides of a coin. Each conditions the other. Platform-as-a-service and core programming are defined in proper wiring programming as a combination. Everybody has a different general business idea. Whatever model it paid for, allocated computing will satisfy the intended customers. Since cloud computing and networking are the two basic architectures around which the cloud is built, Internet access and infrastructure become necessary. That is how institutions often make use of CC and many others.

Although such amounts of data are spread over servers and computers, the currently emerging network is all able to access them with the help of these virtual networks [4]. Accordingly, more application service providers are using infrastructural leasing from infrastructure providers and gap is visible between actual use and operation of equipment needed. For instance, Force Square has abused excellent service of the Amazon EC2 Architecture by billions of days, which brings costs down to about 53% and becomes best cloud resource for denoting measurable requirements.

Many heuristic methods are developed to solve above. These include fair scheduling, sample packing techniques, and first fit, among others. Some indicate meta-heuristic algorithms, such as ant colony and genetic algorithms, exist [5]. The success of these heuristic algorithms depends on manual testing and calibration, besides being influenced by resource demand behavior. This means that it has nothing much to quickly act with further changes in the environment. Areas that rely heavily on machine learning (ML) are computer vision, pattern recognition, and bioinformatics. With the advent of machine learning techniques, large computing systems have been growing.

Google has recently published a report on its efforts to optimize electricity use, minimize costs, and maximize productivity. Machine learning (ML) has revealed to be quite a promising approach to dynamic resource scaling, which one can regard as a potential technique for providing a workload forecast of high accuracy and speed-from data-driven approaches with the future insight in application scheduling. Makespan, cost, and resource usage become additional elements of consideration when scheduling. Various researchers have suggested a number of methods of handling load balancing in both a homogeneous as well as a heterogeneous environment. The overall goal of load balancing is the optimized allocation of tasks amongst available resources and reduction in system processing time [6].

1.1 Research objectives

- to suggest a new method for job scheduling as well as resource allocation in cloud computing networks utilising a meta-heuristic machine learning model.
- In this case, a large number of users and clients with virtual machines are connected to the cloud network.
- Firefly harmony search cat optimisation based on convolutional transfer graph proximal policy is used to implement the task scheduling model for this deployed network. Then, a software-defined virtual machine-based reinforcement markov model is used to allocate the resources.

The organization of this paper is as follows, section 2 explains literature review, proposed model is shown in section 3, section 4 explains resource allocation in proposed model, the experimental results are added in section 5 and section 6 concludes the paper with future scope.

2. Literature review

Many researchers are interested in the task scheduling problem. The strategies can mainly be classified into two broad categories depending on the approach followed in job scheduling: one is the classical path, which includes meta-heuristics and heuristics; in this approach, Work [7] proposed a modified round-robin resource allocation method to minimize waiting times and meet customer requirements. The DGLB [8] reduces energy consumption in data centers by devising energy-efficient as well as regionally load-balanced methods for data center networks. In this all-inclusive scheme, the newer elements of the smart grid like energy storage units are included for handling renewables, incentive pricing mechanisms serve as the design tool, workload and power balancing schemes are set up in network. This is in the endeavor to improve QoS criteria in a geographically dispersed cloud environment- author [9] suggested a linear programming method to solve the web service composition issue named "[illegible]" which chooses the most effective service for each request. Based on control monitor-analyze-plan-execute (MAPE) loop paradigm, an autonomic resource provisioning methodology has been suggested by [10].

Work [11] defined the problem of energy & performance-efficient resource management as a Markov decision process and suggested a unique dimensionality reduction technique. These [12] authors elaborated the PSO-COGENT algorithm; it is much similar to the functioning of the particle swarm algorithm and optimizes time and cost of execution while reducing energy consumption of cloud data centres. To control search procedure against premature as well as divergence issues of PSO, APSO-VI technique offers the nonlinear ideal average velocity. The scheduling method AIRL developed by the authors in [13] is for time-sensitive applications in the cloud through reinforcement learning. The primary objectives are reducing response times and increasing ratios of user requests that the system can satisfy.

The results for AIRL are then compared to several other schedulers including DQN, RR, earliest and random, and according to simulation results, AIRL consistently outperforms these baseline techniques. Scheduling metrics under consideration include response time, success rate, costs of virtual machines, and QoS characteristics for the scheduling paradigm as presented in [14]; this is run in the DQN model based on reinforcement learning. Results from simulation experiments indicate that DQNs do perform better in the respect of the aforementioned parameters than the algorithms cited. The entire experiment was carried out on a live cloud and compared against random, round-robin (RR), and earliest eligible schedulers. [15] The scheduling architecture developed in this framework shortens task execution and waiting times. The authors applied CDDQLS, a reinforcement-learning technique used in machine learning. The whole simulation was run under CloudSim with resource and time constraints. Post simulation, CDDQLS was compared with Random, Time-Sharing, and Space-Sharing algorithms that tend to show a significant effect against the aforementioned algorithms.

A task scheduling model that seeks to minimize the makespan was presented in [16]. It uses a machine-learning approach, DQN, which makes use of a reinforcement-learning approach to schedule activities. The simulations were performed in MATLAB for comparison with the HEFT and CPOP methods. Obtained results show a very substantial reduction in makespan as compared to the baseline methods in question. An algorithm for load-balancing and decision-making that disregarded job sizes was suggested in this paper by the authors [17]. While completing queries, the authors did consider the refreshes on the servers. Work [18] presents the first report on task scheduling based on a vacation queuing model. This approach does not show the way to utilize resources well.

Task scheduling proposed in [19] considers bandwidth as an entity that can be treated as a resource. Nonlinear programming method is developed for allocation of resources to tasks. Rolling horizon scheduling method [20] was invented in real time scheduling for works. The

authors have shown connection between task scheduling as well as energy conservation using resource allocation. Scheduling for concurrent workloads was proposed by Work [21]. In presence of resources, the authors have executed jobs following the FCFS line. The technique of proposed is not primarily concerned with starvation or termination of jobs. [22] suggested a method to rely on Max-Min based scheduling for job execution time forecasting and cloud load balancing. Then jobs were allocated to the VM using the suggested Max-Min approach.

A decrease in response time for the VM and an increase in resource usage followed. Then came the Enhanced Load balanced Min-Min (ELBMM) method for scheduling the static tasks, which was proposed in Work [23]. Tasks were allocated to VMs according to execution time and rescheduled for the purpose of alleviating idle resources. For minimizing makespan and cost combinations for tasks (MCTE) in smart grid-cloud systems, a smart task scheduling is proposed. Such work scheduling in grid-cloud was then mathematically modeled. It was a two-step process. First, choose the largest virtual machine with the highest share of computational resources. The second step is to assign remaining works to VM that can execute them in the shortest possible time. Results showed that RALBA was able to minimize makespan.

3. Hybrid model for task scheduling and resource allocation

The cloud resource scheduling frame based on proposed model in this research is shown in Figure 1. A diverse collection of servers is modeled in private data centers: some feature a low CPU frequency, and are thus more energy efficient in contrast with rather fast systems consuming considerably higher resources, and so on. In public cloud services, use the container orchestration technologies provided by CSPs, such as Microsoft Azure Containers and AWS Elastic Container Services (ECS). This provides the user with the ability to scale down to zero and dynamically provision without having any costs incurred. Our efforts are directed towards automating the workload shifting process in order to optimise renewable energy usage and public cloud expenditures, with the promise of satisfying very high-quality service (QoS) requirements (deadline). Achieving this goal by strategically allocating tasks to various servers in accordance with energy availability and time restrictions. Deploying a public cloud service provider in a dynamically on-demand provided manner to meet the QoS goal in the absence of sufficient computing or renewable energy capacity. The proposed model is as shown in Figure 1.

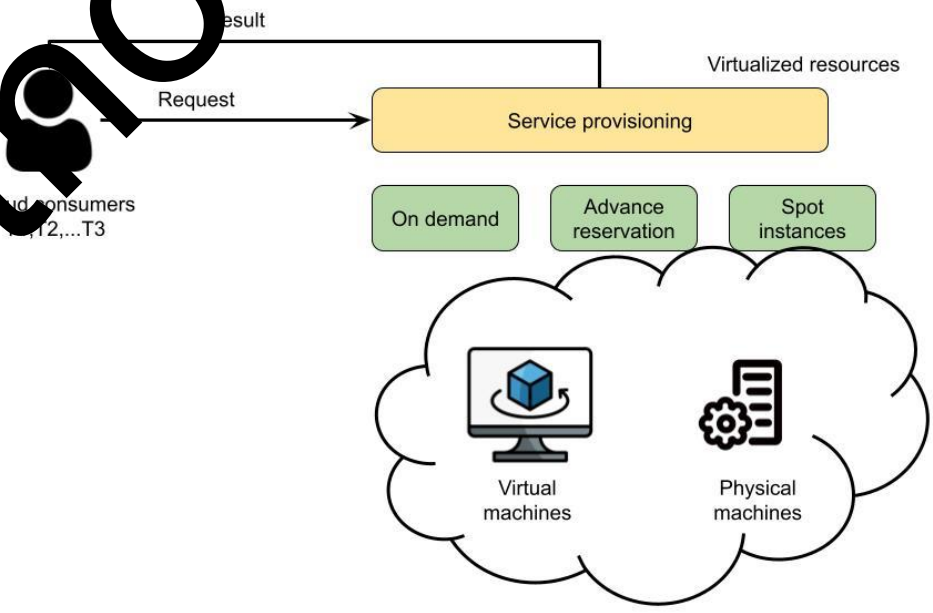


Figure 1. Proposed Hybrid Model for Task Scheduling and Resource Allocation

Task scheduling and server selection are the two types of scheduling in Cloud Computing. Purpose of task scheduling optimization is to give importance to different types of performance measures like cost, execution time, make span, latency, and bandwidth utilization. The objective functions of optimization issues differ from each other based on application and organizational goals-such as optimizing data transfer, minimizing job completion time, makespan, or total service cost. Thus, the objective function and its constraints will vary mathematically depending on type of cloud computing system and objective of organization. Task scheduling algorithms are meant to assign virtual machines (VMs) based on VMs' state and user service requirements.

According to equation (1), the cloud system (CS) is made up of PMs, and each machine is made up of virtual machines (VMs).

$$CS = \left[PM_1, PM_2, \dots, PM_i, \dots, PM_{Npm}\right] \quad (1)$$

The following is an expression of the cloud's PM performance by eqn (2)

$$PM = [VM_1, VM_2, \dots, VM_k, \dots, VM_{Nvm}] \quad (2)$$

When k is between 1 and Nvm, the kth VM is represented as VMk. Nvm is number of VMs, and VMk is kth device of a cloud virtual machine. The following formula is used to determine the VMk features by eqn (3)

$$VM = [SIDV_i, mips_i] \quad (3)$$

SIDTi represents the ith task's identity number, and task-lengthi represents the task's length. Time ECTi is the ith task's completion time, while LIi represents the task preference in terms of the number of tasks Ntsk.

3.1 Convolutional transfer gradient proximal policy-based firefly harmony search cat optimization (CTGPP-FHSCO)

A number of parameters from various service providers, including user request, task type, task dependency, etc., are used to optimise task scheduling process. User request, which comprises of 1 to N tasks units, is first set up. Task type, which consists of 1 to t tasks, is then defined. Total number of tasks in task unit is denoted by term tmax.

Convolutional layer Here is a convolutional layer in one of the CNN core layers. These layers contain filters and are smaller, but they change to encompass the full image. By computing dot product between filter and image, convolutional mechanism operates. The filter region summarises the dot products between the image and filter by eqn (4)

$$a_k^m = \lambda\left(y_j^{m-1} \times x_{jk}^{(1)m} + c_k^{(1)m}\right) \quad (4)$$

Pooling Layer Down sampling was used by the pooling layer. There are various classes of pooling function. Maximum pooling functions are the most often used. The maximum values for each subregion were obtained using the maximal pooling filters. The size feature 4 × 4 × 1 produced a 2 × 2 × 1 size feature if the 2 × 2 × 1 maximal pooling filters were applied.

Fully Connected Layer Each neurone in completely connected layer is connected to neurones in preceding layer. Then, it is said as follows (5):

$$z_k^n = \lambda\left(a_j^{m-1} \times x_{jk}^{(2)w} + c_k^{(2)m}\right) \quad (5)$$

Softmax Function Layer The softmax function layer is utilized to evaluate probability distributions of the event with different events. The following formula (6) is used to calculate and express how the softmax layer operates:

$$Q(z_k^m) = \frac{\exp(z_k^m)}{\sum_{k=1}^{!} \exp(z_k^m)} \quad (6)$$

Device-generated load often adheres to specific guidelines. End device has several tasks km(t) at start of each time slice, data size of km(t) follows a uniform distribution. A randomly selected value between 0 and 1 that is no greater than task arrival probability is then multiplied by this data size. Table I displays task arrival probability in environment as well as task size range, λ i m(t).

Table I. Parameter Settings.

Parameter	Value
$f_m^{\text{finvice},m\in M}$	2.5G
$f_n^{d/s_s}, n \in N$	41.8G
Mobile end device core	C4
$\lambda_m^i(t), m \in M, i \in I, t \in T$	3.0 1, 10.0 Mbit
$r_m\Delta, m \in M$	4 Mbits
$r_n\Delta, n \in N$	41.8 Mbits
$\rho_{m,m}, m$	0.29 gigacycles per Mbit
$\rho_{n,n}, n \in N$	0.297 gigacycles per Mbits
$\tau_{m,m\in M,i\in I}^i m$	100 time slots (10 s)
Task arrival probability	0.3

Every task k i m(t) has unique information, including its number and task size λ i m(t). The jobs initially arrive at the end device's processing queue. It should be noted that both computation and transmission completion are included in completion of processing step described here by eqn (7)

$$\emptyset_m^f(t) = \left[\max_{r' \in \{0,1,t-1\}} P_{m,j}^c(t') - t + 1 \right] \quad (7)$$

The computation time of job k i m(t 0) is indicated by P c m,i (t 0). Determine P c m,i (t 0), c ∈ 2 ∗ C before the end device is placed in the computing queue x because it observes the state of the queue, including the quantity of jobs and their sizes by eqn (8)

$$P_{m,j}^c(t) = \min\left\{ t + \psi_m^j(t) + \left| x_m^i(t) - 1 \right| \left\lceil \frac{\lambda_m^i(t)}{f_m^m \Delta/\rho_m} \right\rceil + x_m^i(t) * \left\lceil \frac{\lambda_m^i(t)}{r_m \Delta} \right\rceil - 1, t + \tau_m^i - 1 \right\} \quad (8)$$

The index of the task's last scheduling destination edge node is used by edge node to determine whether to process or send job. Graph representation agent uses normalisation to determine a "attention coefficient" by eqn (9)

$$a_{ij} = \frac{\exp\left(\text{LeakyReLU}(a^T[WU_i \| WU_j \| L_{i,j}])\right)}{\sum_{k\in N_i} \exp\left(\text{LeakyReLU}(a^T[WU_i \| WU_k \| L_{i,k}])\right)} \quad (9)$$

In the formula above, k stands for concatenation operation, which splices properties of nodes I and J as well as linkages between them. αij is "attention coefficient" between nodes I and J, indicating relevance of node j to node I. Attention coefficient then updates new features Fi of node I by eqn (10)

$$F_i = \sigma\left(\sum_{j \in N_i} \alpha_{ij}^{k} W^{h} U_j\right) \quad (10)$$

3.2 Proximal Policy (PP)

PP optimizes policies over continuous action spaces. This deep learning method provides for smooth and effective policy updates without causing major, disruptive changes. A built-in characteristic of the method also enables fair resource allocation for jobs with real-time fluctuating workloads. By adjusting the criteria for resource and workload allocation in an optimal sense, PPO improves the model, assuring that no edge or cloud node is ever committed or underutilized in its workload. The PPO procedure is depicted in Figure 2. The complete work follows.

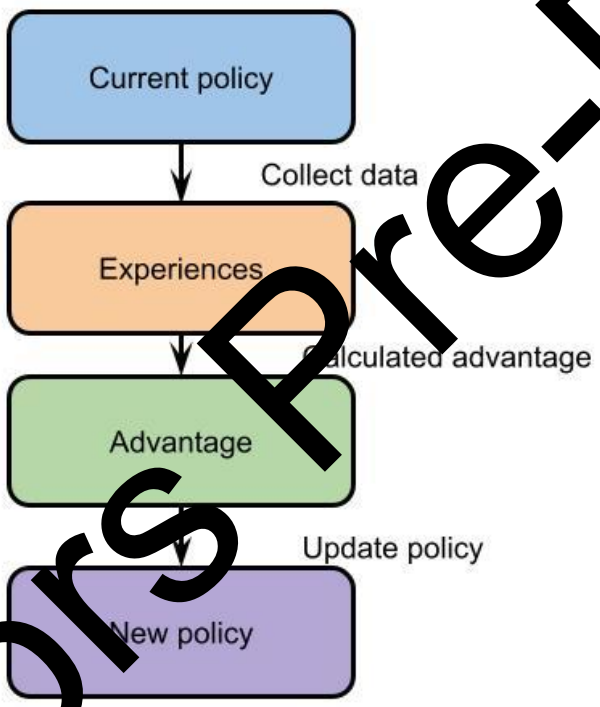


Figure 2. Proximal policy model

- Representation of policy: The policy πθπθ(a|s) determines the probability of acting in a given condition.
- Collection of statistics: Combine experiences by putting current policy into practice and storing results in a buffer.
- Estimation benefits: Determine advantage function. A^(s,a) Evaluate each activity's relative efficacy in proportion to the expected result using Equation (11).

$$\widehat{\mathrm{A}}(s, \mathrm{a}) = [r + \Psi V_{\Phi}(\dot{s}) - V_{\Phi}(s)] \quad (11)$$

- Update policy: Use the reduced aim to optimise the policy and prevent over-updating with Eq.(12)

$$L^{PPO}(£_P) = E\left[\min\left(r_t(£_P)A(s_t, a_t), \mathrm{clip}(r_t(£_P), 1-\mathcal{E}, 1+\mathcal{E})A(s_t, a_t)\right)\right] \quad (12)$$

- Update Value Function: To improve precision of future reward prediction using Eq., refine the value function by eqn (13)

$$L^{V}(\Phi) = E_t[(V_{\Phi}(s_t) - R_t]' \quad (13)$$

3.3 Firefly harmony search cat optimization

To address job scheduling issue in cloud scenarios, technique to be used is called optimization by firefly. These fireflies can be attracted to another firefly depending on how brilliant it is. Obviously, they are attracted to the flies' opposite sex. The intensity of the headlights, which is the mode of communication, determines the sex of the fly. Generally, the following conditions are kept in view for firefly optimization technique: (1) Assumed all are to a single sex of flies, and they are attracted regardless of sex. (2) Brightness is equal to attraction, which means a less bright firefly will attract to a brighter one. (3) A fly goes at random if its brightness is greater than that of any other fly in the search space. Because appeal is directly correlated with brightness, the distance between flies increases if their light is less strong, or brighter. Therefore, compute the brightness—that is, the intensity and attractiveness—in order to move further with this strategy.

Initially, Equation (14) can be used to determine a firefly's intensity.

$$Int(s) = \frac{Int(r)}{s^2} \quad (14)$$

The brightness and the way light is absorbed determine how attractive something is. Consequently, there is a relationship between the brightness intensity and the light absorption coefficient. This is what Equation (15) shows.

$$Int = Int_0 \cdot c^{\gamma} \quad (15)$$

First determine the distance between two fireflies after determining their intensity. Equation (16) is utilised in the computation.

$$s = |x^t - x^d| = \frac{1}{r}\sqrt{\sum_{i=1} (x^{(0)2} - x^{(t2)})^2} \quad (16)$$

Determined the population of fireflies and scattered them after calculating the distance. Movement of a firefly ii that is attracted to jj, a firefly that is brighter than ii, for the number of iterations denoted by u. Equation (17) is used to compute the firefly's movement.

$$x_{u+1}^d = x_u^d + r_0 e^{-\left\{\frac{s^2}{d^2}\right\}}(x_u^d - x_u^d) + \gamma EUR_t \quad (17)$$

Step 1: Setting up the optimisation algorithms, initialising the number of solutions is a crucial step. Here, set parameter for this harmony search method to its initial value. This approach uses a number of parameters, including Pitch Adjusting Rate (PAR); ∈[0,1], Harmony Memory Size (HMS), Harmony Memory considering Rate (HMCR), and HMCR; ∈[0,1].

Step 2: Set up Harmony Memory (HM) initially. In this case, harmony memory HM, which is provided by Eq. (18), is produced arbitrarily.

$$HM = \begin{bmatrix} HM_1^1 & HM_2^1 & \dots & HM_n^1 \\ HM_1^2 & HM_2^2 & \dots & HM_n^2 \\ HM_1^{HMS} & HM_2^{HMS} & \cdots & \dots \end{bmatrix} \quad (18)$$

Step 3: Create a brand-new harmony. Three rules, such as memory, pitch change, and random selection, are taken into consideration when creating a new harmony vector. Creating a new harmony is what improvisation is all about. Any of qualities in preset (HM_1^1-HM_1^HMS)

range can be used to estimate the major choice variable HM1 1 for the new vector in the memory consideration by eqn (19)

$$HM_i^{\text{New}} = \begin{cases} HM_i^{\text{New}} \in \{HM_i^1, HM_i^2 \ldots \text{HM}_i^{\text{HMS}} \text{ with probability } HMCR \\ HM_i^{\text{New}} \in HM \text{ with probability } (1 - HMCR) \end{cases} \quad (19)$$

Pitch adjustment is given in Eq. (20) as follows:

$$HM_i^{\text{New}} = \begin{cases} \text{Adjusting pitch with probability } y \\ \text{Doing nothing with probability } (1 - PAR) \end{cases} \quad (20)$$

In the event that HMNew i's pitch adjustment decision is YES, HMNew i is changed as follow (21):

$$HM_i^{\text{New}} = HM_i^{New} \pm rand \times b \quad (21)$$

Step 4: In order for this to occur, all cats are first split into two grou[illegible] seeking and active modes, and then they are randomly initialised by developing s[illegible]arm. A fitness value must be determined for each iteration for each cat that is in active mode. Equation (22) below is used to determine the velocity for each cat after they have been initialised.

$$\text{ve}_d^q(t+1) = s * ve_d^d(t) + b * u * \left(x_{\text{ded}}^d - x_q^d\right) \quad (22)$$

where x d best is the best solution for that iteration, u is a rand[illegible] value between 0 and 1, b is a constant, and ve q d (t) is velocity of qth cat at the tth iteration. Until all iterations have been finished, the velocity and location updates of the cats must be calculated.

Algorithm of CTGPP-FHSCO

1. Input: State of device m in the slot t, $m \in M$: $S_m(t) = E_m(t), U(t), H(t), \text{Link}(t)$
2. Output: Action chosen for every task;
3. Obtain $H'(t)$ by passing $H(t)$ in each GRU;
4. Obtain $F_m(t)$ by passing $H'(t)$ and $\text{Link}(t)$ in graph represent agent;
5. Forward $F_m(t)$, $E_m(t)$ and $H'(t)$ to scheduling agent;
6. Obtain $H'(t)$ by passing $H(t)$ in GRUs of te scheduling agent;
7. for Every task $E_m(t)$ do
8. Obtain $V_{m,i}(t)$ by concatenating $E_{m,i}(t), H'(t)$ and $F_m(t)$;
9. Evaluate state value $V_{m,i}(t)$ and advantage_value $A_{m,i}(t)$;
10. Evaluate Q value $e_{m,i}^{\text{sche}}$;
11. Obtain $\pi_{m,i} = \text{Softmax}\left(Q \text{ value } {}_{m,i}^{\text{sche}}\right)$
12. end for
13. $\forall k, t, m: \gamma_{ikm} = 0$.
14. $\forall j, m, m', i, k: \delta_{j(m,m')}^{k(i,i+1)} = 0$.
15. for $k \in K$ do
16. for $i \in I_k$ do
17. if i is the first subtask in I_k then
18. $m' = \arg\min_x OC_{ikx}$.
19. Else
20. $m' = \arg\min_x \left[OC_{ikx} + LC_{j(m,x)}\right]$
21. $\delta_{j(m,m')}^{k(i,i+1)} = 1$
22. end if

23. $\gamma_{ikm'} = 1$
24. $m := m'$
25. end for
26. end for

4. Software defined virtual machine based reinforcement markov (SDVM-RM) model

Through centralized control and adaptive dynamic behavior on the network, Software Defined Networking (SDN) enhances cloud network resource allocation by enabling efficient virtual machine (VM) provisioning and management. This means that the operation of network infrastructure can be done using software rather than hardware-based setup, thus giving VMs in a cloud environment flexibility and responsiveness in resource allocation. The system model being considered is for cooperative virtual machine and bandwidth allocation in the cloud service provider and ISPs. Users will submit demand requests for virtual machine provisioning to a central controller. One significant service paradigm that allows cloud providers to give cloud users access to enormous computing capabilities via the Internet is Infrastructure as a Service (IaaS). This issue is further compounded by price fluctuations, demand uncertainties, and other issues. Therefore, the VM allocation method needs to be optimised to fulfil resource utilisation requirements and minimise user charges. This issue is known as VM allocation optimisation, and Figure 3 illustrates it.

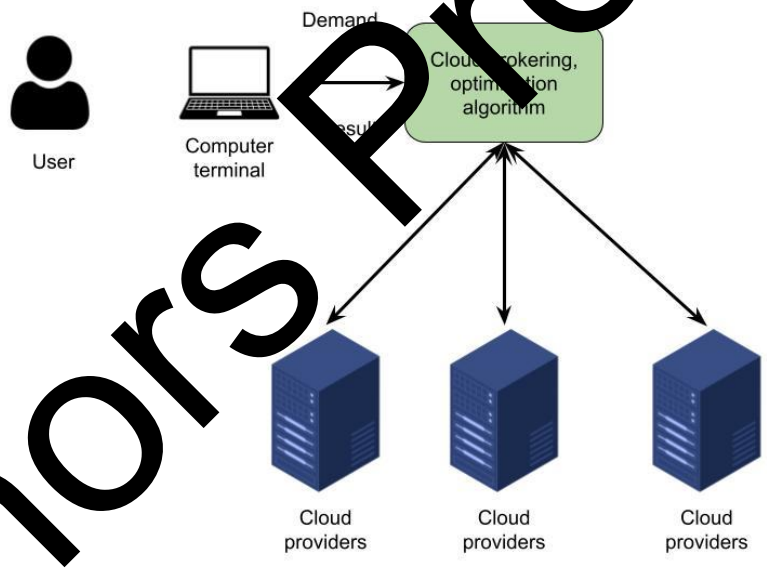


Figure 3. VM resource allocation in cloud computing

Controller then procures required bandwidth virtually through ISPs and the required VMs from cloud providers. ISPs can deploy their SDN through OpenFlow-enabled switches so that the centralized controller can allocate bandwidth and route traffic through virtual routers. Model the decision-making process with respect to a single controller but would ideally work with many controllers, thus managing networks and providers. In the reservation phase, bandwidth and VM are reserved well ahead of time-in most cases, a year-before their real requirement becomes clear. The second phase occurs when the real demand of the users—such as the daily demand-is realized at actual use. Utilisation and on-demand phases are the two divisions of the second stage. The necessary, reserved resources are used during the utilisation phase, typically at a little cost. The algorithm moves into on-demand phase if real demand exceeds resources that have been reserved. To meet any unmet demand during on-demand

phase, more resources may be provisioned at a higher cost. Since the first stage is far less expensive than the second but less flexible because of the lengthy reserve period, it is crucial that it be decided as best as possible. R = {1,...,R}, where R is the total number of routers, represents the collection of virtualised routers that an ISP oversees. While all expenses are known ahead of time, the demand for virtual machines is unknown. Thus, the collection of all potential VM demand values are provided by eqn (23)

$$\mathcal{D} = \prod_{V_i \in \mathcal{V}} \mathcal{D}_i = \mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_{|\nu|} \quad (23)$$

The total bandwidth required at the time of reservation is also unknown due to the erratic demand for virtual machines. The set of potential bandwidth requirements for class Vi can be obtained using d(b) i, (1), since each VM class has a fixed external bandwidth demand. The following defines the range of potential bandwidth needs for VM class Vi by eqn (24)

$$\mathcal{B}_i = \mathcal{D}_i \cdot d_i^{(b)} \quad (24)$$

The collection of potential external bandwidth needs for each virtual machine class is thus represented as follows by eqn (25)

$$\mathcal{B} = \prod_{V_i \in \mathcal{V}} \mathcal{D}_i \cdot d_i^{(b)} \quad (25)$$

Finding a policy π that increases total reward R when method transitions between states following specific MDP stages T is aim of solving this MDP. Equation (1) determines the total reward R, where ri represents reward of every time step i and γ is discount factor (0 < γ < 1, in this article set to 0.9) that prevents the total reward from reaching infinite by eqn (26)

$$R = \sum_{i=1}^{T} \gamma^{i-1} r_i \quad (26)$$

The predicted total reward for an agent beginning in state s with policy π is then represented by value function of every state V π (s), which is defined in Equation (2). V π (s) thus shows how favourable state s is for an agent to remain in. There is an ideal policy π ∗ among all the others that maximises V π (s), as indicated by Equation (27).

$$V^*(s) = E\left[\sum_{i=1}^{T} \gamma^{i-1} r_i\right]$$

$$\pi^* = \arg\max_{\pi}^{*} V^*(s) \quad (27)$$

Then, for simplicity's sake, define V (s) as abbreviated form of V π (s). In reinforcement learning, the agent must test every policy π, which includes every conceivable combination of state-action pairings (s, a), in order to obtain the optimal V (s). Thus, for all conceivable actions a's, maximum value of Q(s, a) (Q∗ (s, a)) equals maximum value of V (s) (V ∗ (s)) (Equation (28)).

$$V^*(s) = Q^*(s,a) = \max Q(s,a) \quad (28)$$

$$Q^*(s,a) = \sum_{s'} P(s' \mid s,a) r(s,a,s') + \gamma \sum_{s'} P(s' \mid s,a) V^*(s') \quad (29)$$

Thus, for every feasible state s 0 that transits from state s taking action a, derive Equation (6) using Equations (4) and (5). Positive or negative rewards are kept in the replay memory. It should use equation 21 to update its state space to the following state after considering incentives. This process keeps on until the final state space, or task, is reached as depicted in Figure 4.

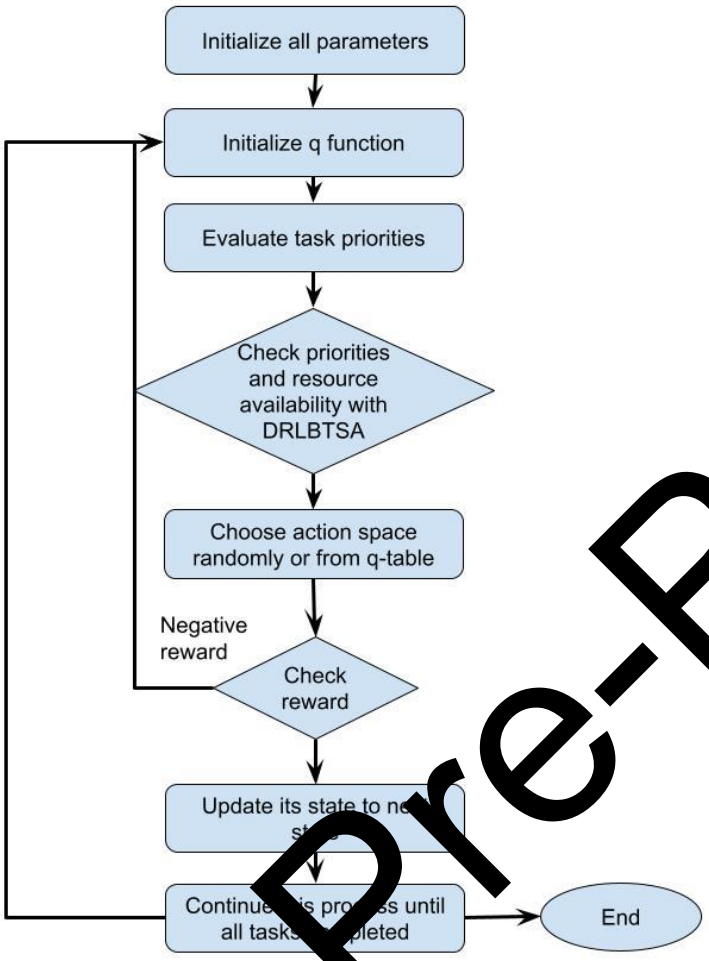


Figure 4. flowchart for SDVM-RM

5. Results and Discussion

A cloudlet simulator is used to replicate the suggested model, and test outcomes are assessed to gauge how well it performs. A number of parameters, including resource utilisation, acquisition speed, execution time, energy management, are examined in light of the data produced. Build a cloud data centre with continuous PM measurements. Additionally, it begins using resource agents to create data canters. Every data centre began with several data hosts and related virtual machines. The hardware and simulation settings are displayed in Tables II and Table I

Table II. Hardware Specifications.

Required	Component Specification
Processor	Intel ® Pentium ® CPU G2030@ @3.00GHZ
RAM	4 GB
Hard Disk	1 TB
Operating System	Windows (X86 ullimate) 64-bit OS
System	64 Bit OS System

Table III. Simulation Settings.

Component	Specification	Values
Cloudlets	Length of task	1600-3400
	No of tasks	30-300
Virtual Machine	Host	4
	Memory	540
Physical Machine	Bandwidth	25,00,00
	Storage	500 GB

Assessing performance of proposed method entails specifics such as power consumption, data centre resource utilization, acceptance rate and time to implement. Implement technology for official site visits using block processing approach. Jobs arrive at t = 0, then provide a distribution system that passes it through. Our framework work planning idea prioritizes jobs. In particular, resource agents that will allocate resources from the resource table get priority. Tasks assigned to virtual machines employ the space-sharing strategy that maximizes resource utilization in method. Performed ten iterations of each experimental evaluation for various QoS parameters to ensure the reliability of our results, recorded the mean outcomes to remove any inconsistencies. Because each simulation ran for 30 minutes, performance data could be thoroughly analyzed.

5.1 Parameters for VMs, datacentres

This architecture setup three various types of data centers-fog data centers, cloud data centers, and hybrid data centers. Each of these data centers consists of numerous servers and varies with Fog nodes and virtual machines. The space-sharing policy, maximising system resource usage, is employed to assign workloads to virtual machines. To ensure the results' reliability, each case with varied QoS parameters was tried ten times, and the means were taken to eliminate discrepancies. Each run was scheduled for 30 minutes of simulation execution, so performance data could be scrutinized in depth.

Table IV presents the detailed infrastructure and configuration of the data centers. The three data centers have a total of twelve hosts- which means a perfect fit to house a grand total of sixty users. Each of the hosts has been designed with a capacity of 64 GB RAM, 10 TB of storage to meet the requirements of even the most demanding applications. This would be connected by a network with extremely high bandwidth of 150 GB/s, enabling very fast data transfers, and making space-sharing based on dynamic assignment possible in real-time for flexible resource management according to demand. Each server capability consists of twenty CPU cores and has a network latency of less than three milliseconds. Hence multiple concurrent loads can be effectively handled with a minimum processing delay.

Table IV. Data center and host configuration

Cloud Entity	Characteristic	Value
Data Center	Number of Data Centers	3
	Number of Users	60
	Number of Hosts	12
Host	Storage Capacity	10 TB
	Shared Policy	Space Shared with Dynamic Allocation
	Bandwidth (BW)	150 GB/s
	RAM	64 GB
	CPU Cores	20 Cores
	Virtualization Technology	KVM
	Power Consumption	15 kW
	Network Latency	3 ms

Figure 5 (a)-(e) shows a graphic representation of resource utilization, network efficiency, throughput, latency, QoS. Of these, the memory and a particular work processor are some resources' utilization. Whenever the other two methods are combined, will result in increased users using a particular method. The amount of resources utilized with the application of different resource allocation strategies. It expresses that the resource utilization percentage of the proposed method reaches its peak for a range of working sizes; being, it is functioning like that of smart city systems.

The proposed scenario agrees with real-time situations in which failures of the server or network congestion could lead to baited changes in resource availability. In this situation, there are 6,000 jobs, with all jobs being 100 (MI) in length. In a cloud system, the amount of disparity in the workload assigned to each computing resource is referred to as the degree of imbalance. Less imbalance in an algorithm ensures better fair work distribution across resources such that none is overworked relative to others being underutilised. This aspect becomes imperative in cloud computing since it is a great influencer of the model's efficacy and performance. Until more than 400 episodes of training, not very great optimising takes place; after that, the curve started converging slowly in the forward direction. The above graph shows average task time, the variations in work duration, and the energy consumption under weight settings (0.8, 0.6, 0.4, and 0.2). In this way, the proposed algorithm can successfully balance task makespan and energy consumption by varying the weights of different goal reward functions, as evidenced by the curve in the image. In general, with task makespan in consideration, the system would adopt a strategy of opening more servers or increasing demand on the server to reduce wait time for tasks.

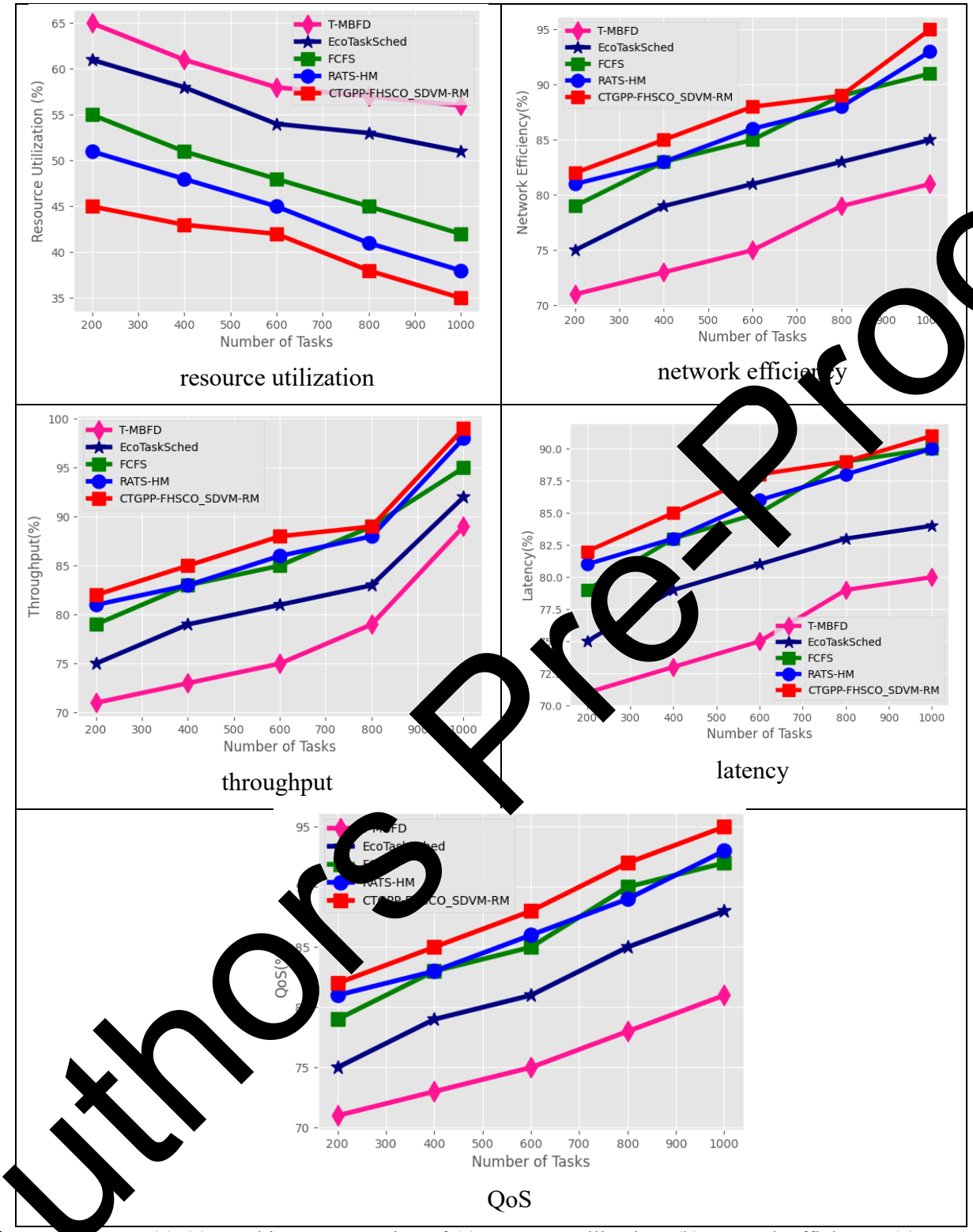


Figure-5 (a)-(e) graphic representation of (a) resource utilization, (b) network efficiency, (c) throughput, (d) latency, (e) QoS for proposed and existing technique

The energy consumption of the system would increase as a result of overburdened and wasted server resources. In contrast, if the optimization target were minimal energy consumed, then the technique would adjust the resource utilization of the server to optimal utilization for minimizing global energy consumption. Response time, in essence, is the time elapsed by the system while considering a particular request. It can also be stated that the availability of resources directly affects reaction time. There are task scheduling methods to allocate

resources. If task scheduling is done appropriately, response times can be reduced since resources would inherently be available earlier or before deadlines. The majority of existing systems do not consider bandwidth, which is considered a critical resource. Bandwidth happens to be one of the three considerations of cloud computing data centres that advocate for in this paper. In this particular case, consider a set of nine different tasks from a Google task events dataset to feed them into five different scheduling algorithms and thus concretely illustrate how effective the new method is. Every task is identical to the one already used.

For reinforcement learning, the average service latency is essentially governed by how well the learning agent knows its environment. Positive results would be produced through better exploitation of resources and examination: these two must coexist. The ε value determines preference on using exploration or exploitation. Therefore, choose to exploit ε=0.5 for our method. The proposed algorithm has ε value-configured actions. A random value from the q-table is selected whenever a randomly chosen integer is less than ε (0.5) to determine the q-value. The algorithm's performance would be lessened by not selecting an action that is highly rewarded; that is, an optimal action. The delay in the other processes queued behind it means that the task is being assigned to a slower virtual machine. This explains why a high value for ε would yield a delay. Conversely, lower values of ε would also imply a lower probability that the random number would be less than ε. When applying Equation 2, this aids the learning agent in selecting the maximum q-value. This reduces service latency and enables the best possible use of virtual machines. Nevertheless, the learning agent is limited in its ability to explore the environment and is compelled to execute Equation 2 in each iteration when the ε value is 0. This gets rid of exploration, which leads to bad virtual machine selection and VM queue congestion. Therefore, selecting an ε value that can lead to effective management between exploration and exploitation is crucial. Because of this, naturally decided to choose ε = 0.5 for our strategy.

6. Conclusion

Using a meta-heuristic machine learning model, this study suggests a new method for task scheduling as well as resource allocation in cloud computing networks. Numerous users and clients with virtual machines are utilising the cloud network in this instance. This deployed network's task scheduling model is implemented utilising firefly harmony search cat optimisation based on convolutional transfer graph proximal policy. Next, a reinforcement markov model based on software defined virtual machines is used to allocate resources. Multi-tenancy makes scheduling in the cloud model an extremely dynamic scenario because different workloads need resources according to the processing capacity for demands. The proposed heuristic strategy effectively distributes the resources of high value taking into consideration resource utilization. On the metrics of optimal deployments of computational resources: CPU, memory and bandwidth could be determined. Throughput of 97%, latency of 95%, QoS of 98%, network efficiency of 96%, and resource utilisation of 45% were all achieved using the suggested method. Proposed system adds a resource-bandwidth for performance assessment as opposed to most existing systems that consider workloads on CPU and memory resource usage. Future research in this area will focus mainly on developing more efficient scheduling methods that will improve response and turnaround times.

Conflict of interest: The authors declare no conflicts of interest(s).

Data Availability Statement: The Datasets used and /or analysed during the current study available from the corresponding author on reasonable request.

Funding: No fundings.

Consent to Publish: All authors gave permission to consent to publish.

Reference:

[1] Gurusamy, S., & Selvaraj, R. (2024). Resource allocation with efficient task scheduling in cloud computing using hierarchical auto-associative polynomial convolutional neural network. *Expert Systems with Applications*, *249*, 123554.

[2] Alla, V. R. S. P., Medikondu, N. R., Parige, L. S., Satyanarayana, K., Kankhva, V. S., Dhaliwal, N., & Saxena, A. K. (2024). Optimizing task scheduling in cloud computing: a hybrid artificial intelligence approach. *Cogent Engineering*, *11*(1), 2328355.

[3] Mangalampalli, S., Karri, G. R., Kumar, M., Khalaf, O. I., Romero, C. A. T., & Sahib, G. A. (2024). DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing. *Multimedia tools and applications*, *83*(3), 8359-8387.

[4] Komarasamy, D., Ramaganthan, S. M., Kandaswamy, D. M., & Mony, G. (2025). Deep learning and optimization enabled multi-objective for task scheduling in cloud computing. *Network: Computation in Neural Systems*, *36*(1), 79-108.

[5] Kaur, S., Singh, J., & Bharti, V. (2024, March). A Comparative Study of Optimization Based Task Scheduling in Cloud Computing Environments Using Machine Learning. In *2024 5th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)* (pp. 731-740). IEEE.

[6] Rajawat, A. S., Goyal, S. B., Kumar, M., & Malik, V. (2025). Adaptive resource allocation and optimization in cloud environments: Leveraging machine learning for efficient computing. In *Applied Data Science and Smart Systems* (pp. 499-508). CRC Press.

[7] He, H., Gu, Y., Liu, Q., Wu, H., & Cheng, L. (2025). Job Scheduling in Hybrid Clouds With Privacy Constraints: A Deep Reinforcement Learning Approach. *Concurrency and Computation: Practice and Experience*, *37*(1), e8307.

[8] Devi, N., Dalal, S., Solanki, K., Dalal, S., Malhore, U. K., Simaiya, S., & Nuristani, N. (2024). A systematic literature review for load balancing and task scheduling techniques in cloud computing. *Artificial Intelligence Review*, *57*(10), 276.

[9] Wang, X., Laili, Y., Zhang, L., & Liu, Y. (2024). Hybrid task scheduling in cloud manufacturing with sparse-reward deep reinforcement learning. *IEEE Transactions on Automation Science and Engineering*.

[10] Sharma, S., & Rawat, S. S. (2024). Efficient resource allocation in cloud environment using SHO-ANN-based hybrid approach. *Sustainable Operations and Computers*, *5*, 141-155.

[11] Dahiya, M., Kaur, C., Bisht, A. S., Nimma, D., Dhanalakshmi, G., & Faizal, M. M. (2024, December). Hybrid Deep Learning Framework for Dynamic and Energy-Efficient Workload Migration in Cloud Computing Environments. In *2024 International Conference on Communication, Control, and Intelligent Systems (CCIS)* (pp. 1-6). IEEE.

[12] Javien, H., Javadpour, A., & Sangaiah, A. K. (2024). Efficient task scheduling in cloud networks using ANN for green computing. *International Journal of Communication Systems*, *37*(5), e5689.

[13] Malti, A. N., Hakem, M., & Benmammar, B. (2024). A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems. *Cluster Computing*, *27*(3), 2525-2548.

[14] Mahdizadeh, M., Montazerolghaem, A., & Jamshidi, K. (2024). Task scheduling and load balancing in SDN-based cloud computing: A review of relevant research. *Journal of Engineering Research*.

[15] Sefati, S. S., Nor, A. M., Arasteh, B., Craciunescu, R., & Comsa, C. R. (2025). A Probabilistic Approach to Load Balancing in Multi-Cloud Environments via Machine Learning and Optimization Algorithms. *Journal of Grid Computing*, *23*(2), 1-36.

[16] Afzali, M., Mohammad Vali Samani, A., & Naji, H. R. (2024). An efficient resource allocation of IoT requests in hybrid fog–cloud environment. *The Journal of Supercomputing*, *80*(4), 4600-4624.

[17] Sandhu, R., Faiz, M., Kaur, H., Srivastava, A., & Narayan, V. (2024). Enhancement in performance of cloud computing task scheduling using optimization strategies. *Cluster Computing*, *27*(5), 6265-6288.

[18] Amini, P., & Kalbasi, A. (2024, May). An adaptive task scheduling approach for cloud computing using deep reinforcement learning. In *2024 Third International Conference on Distributed Computing and High Performance Computing (DCHPC)* (pp. 1-9). IEEE.

[19] Alsubaei, F. S., Hamed, A. Y., Hassan, M. R., Mohery, M., & Elnahary, M. K. (2024). Machine learning approach to optimal task scheduling in cloud communication. *Alexandria Engineering Journal*, *89*, 1-30.

[20] Khademi Dehnavi, M., Broumandnia, A., Hosseini Shirvani, M., & Fathian, I. (2024). A hybrid genetic-based task scheduling algorithm for cost-efficient workflow execution in heterogeneous cloud computing environment. *Cluster Computing*, *27*(8), 10833-10858.

[21] Mangalampalli, S., Hashmi, S. S., Gupta, A., Karri, G. R., Kodukumar, K. V., Chakrabarti, T., ... & Margala, M. (2024). Multi objective prioritized workflow scheduling using deep reinforcement based learning in cloud computing. *IEEE Access*, *12*, 5373-5392.

[22] Jeon, J., Park, S., Jeong, B., & Jeong, Y. S. (2024). Efficient container scheduling with hybrid deep learning model for improved service reliability in cloud computing. *IEEE Access*.

[23] Pachipala, Y., Dasari, D. B., Rao, K. V. K. M., Bethapudi, P., & Srinivasarao, T. (2024). Workload prioritization and optimal task scheduling in cloud: introduction to hybrid optimization algorithm. *Wireless Networks*, 1-20.