Contemporary Mathematics
http://ojs.wiserpub.com/index.php/CM/

UNIVERSAL WISER PUBLISHER

Research Article

CSO-based Efficient Resource Management for Sustainable Cloud Computing

K. Shanmugam*, Satyam K, T. Rajasekhar

Department of Computer Applications, Annamacharya Institute of Technology and Sciences, Karakambadi, Tirupathi, Andhra Pradesh 517520, India
Email: karnam.shanmugam2019@gmail.com

Received: 16 March 2023; **Revised:** 3 April 2023; **Accepted:** 6 July 2023

Abstract: The pervasive need for cloud-hosted application services has resulted from the widespread use of cloud data centers. Not only that, but there has been a dramatic increase in the resource demands of current applications, especially in data-intensive businesses. This has resulted in an increase in the number of cloud servers made available, which has increased energy usage and prompted environmental concerns. Only partially do the difficulties of scalability and adaptability in cloud resource management get addressed by conventional heuristics and reinforcement learning-based techniques. Many existing works overlook the interdependencies between host temperature, task resource usage, and scheduling decisions. Especially in contexts with fluctuating resource demands, this results in poor scalability and an upsurge in computing resource requirements. The study recommended a holistic resource management strategy based on resource scheduling for enduring cloud computing as a solution to these restrictions. Energy, thermal, and cooling models are all taken into account in the proposed model, which expresses the optimization of data center energy efficiency as a multi-objective scheduling issue. To generate optimal scheduling decisions and approximate the quality of service (QoS) for a given system state, the model employs cat-based swarm optimization as a surrogate model. Using the China Ocean Shipping Company (COSCO) framework, we conducted experiments that demonstrate cloud service orchestrator (CSO)'s superior performance compared to state-of-the-art baselines in terms of energy ingesting, makespan, and execution overhead in both simulated and real-world cloud environments.

***Keywords*:** cloud data centers, cat-based swarm optimization, CloudSim, sustainability, QoS, energy efficiency

MSC: 68-04

1. Introduction

Cloud-fog computing is the practice of making use of a cloud provider's networked, off-site hardware components, such as storage and processing power, rather than on-site hardware [1]. Customers can get the computational power they need on demand without investing in any hardware [2]. Organizations can benefit from cloud-fog computing's scalability and adaptability because it's easy to scale up their use of the cloud provider's processing resources as their demands grow. Less reliance on local hardware also minimizes associated expenses and maintenance requirements [3]. As a result, cloud service providers must contend with a larger carbon footprint and higher energy prices. As a result, it

DOI: https://doi.org/10.37256/cm.5220242700

is preferable to restrict these resources' energy intake as much as possible [4]. High energy consumption may also have an effect on service reliability since it is correlated with higher operating temperatures, which in turn can create hotspots that have an adverse effect on performance. In order to make the most of the time, money, and energy benefits of cloud-fog tasks, migration has become an integral aspect of modern task scheduling. To keep the edge servers' computing and processing running smoothly, just a small number of virtual machines (VMs) should be moved from one to another [5]. In a dynamic edge computing environment, the demand for edge servers is constantly increasing. Only if there is no contention for these shared resources can everyone use them without incident. Predicting how much time each VM will need to use the central processing unit (CPU) to complete its activities early on helps in the migration of potentially competing VMs [6]. The tasks may be carried out while still providing the required level of service since resource contention can be anticipated and VMs can be allocated accordingly (quality of service (QoS)). Predicting when resource contention will occur is also crucial to meeting service level agreement (SLA) requirements for applications.

Here, an edge cloud architecture is used, in which operations are executed in parallel on several edge servers, with offloading to the cloud performed only when absolutely necessary [7]. For dealing with the massive amounts of data produced by online gaming platforms and vehicular applications utilized by drivers while on the move, the edge cloud is invaluable. Since multiple edge servers actively process the data fashion, the data edge cloud architecture is fast [8, 9]. In comparison to using multiple, less effective models, the performance of hybrid models is enhanced. Certain models are effective at anticipating the resources required for a lesser number of users with unchanging demand over a set period of time, but they are unable to do so when the number of users suddenly increases. For moderate numbers of users with changing needs, different methods can make accurate predictions of available resources. Under these stipulations, it becomes possible to combine the best features of various models and compensate for their deficiencies by employing a single one [10, 11].

Although cloud computing's robust processing and caching capabilities make it well-suited for handling complicated vehicular computing tasks, doing so inevitably results in significant latency because all requests must be sent to cloud servers before they can be processed [12]. Furthermore, the additional delay brought on by vehicle mobility is not something to be disregarded since it has a direct impact on the efficiency and security of transportation [13, 14]. Shorter communication times in the multi-access edge computing (MEC)-aided Internet of Vehicles (IoV) can be achieved through the development of appropriate transport protocols [15], which will increase connectivity between vehicles. In-network storage and offloading solutions have also been implemented to help mitigate the effects of mobility cars on the network and tackle the associated problems. In addition, deep reinforcement learning (DRL) introduced to MEC-enabled IoV situations can effectively manage network resources to enhance conveyance presentation [16].

Artificial intelligence (AI) may automatically learn and adapt to new environments and provide more nuanced responses based on pre-installed algorithms and computer knowledge built on data analysis. Here, we offer an optimization-based holistic resource organization method for power-saving cloud computing. To maximize efficiency in a hybrid public-private cloud setting, the suggested technique substitutes a cat behavior for QoS metrics. By using the power ratio as a heuristic, we are able to rapidly narrow down the scheduling search space and drastically shorten the scheduling process. Comprehensive studies on both virtual (with the help of the CloudSim toolkit) and real-world (with the help of the China Ocean Shipping Company (COSCO) framework) cloud testbeds demonstrate that the projected model outperforms the current crop of sustainable computing schedulers.

Here is how the rest of the paper is laid out: In Section 2, we provide a short-lived impression of algorithms from several fields that can be utilized for scheduling to save energy. The meat of our work is labeled in Section 3, while Section 4 has a detailed discussion and presentation of our solution models. The results of our trial are presented and discussed in Section 5. The final Section 6 wraps everything up and lays out some recommendations for the road ahead.

2. Literature review

Iftikhar et al. [17] investigate how smart task-scheduling algorithms can be used to distribute user-deployed jobs among servers in order to cut down on energy usage. In the past, algorithms for scheduling tasks have relied on a mixture of heuristic and metaheuristic approaches. As an example, the gated graph convolution network (GGCN) is a recent advancement in the use of AI to optimize task scheduling. To investigate the impact of GGCN's gated recurrent unit being extended to a unit, this article presents a novel strategy named HunterPlus. The research also explores how

convolutional neural networks (CNNs) can be used to better schedule tasks in the cloud and fog. Based on experimental data, the CNN scheduler is superior to the GGCN-based replicas in terms of both energy consumptions per task correspondingly.

To ensure the long-term viability of cloud data centers, Karthikeyan et al. [18] present a tree-structured tree hierarchical deep convolutional neural network (T-CNN) tuned with a sheep flock based on workload data. In the first stage, the kernel correlation technique is used to preprocess the available cloud data center historical data. Workload prediction in an ever-changing cloud setting is made possible with the help of the proposed T-CNN method. The T-CNN model's weight parameters are tuned using the sheep flock optimization algorithm. The suggested COSCO2 approach successfully forecasts future workloads while also decreasing the wasteful use of energy in cloud data centers. Two reference datasets are used to assess the efficiency of the projected method: the (i) Space Agency of the United States of America and (ii) HTTP traces of the province of Saskatchewan. This model's simulation has been built using a Java tool, and its parameters have been computed. The simulation margin For example, when validating the NASA dataset, the proposed method achieves a high accuracy of 20.64%, 32.95%, 12.05%, 32.65%, and 26.54% with a lower energy consumption of 27.4%, 26%, 23.7%, 34.7%, and 36.5%. When validating the Saskatchewan HTTP trace dataset, the proposed method achieves a high accuracy of 2.

Both a dynamic Markov perfect for resource argument prediction in edge cloud (Dynamic Markov model for Resource Contention Prediction in Edge Cloud; DMRCP) and a proposed by Surya and Rajam [19] to servers (Hybrid Cascade of Regression and Markov model for Resource Contention Prediction; CRMRCP). DMRCP revises a history matrix every time a VMs CPU is put to use. The probability matrix for transitions is updated using the history matrix. With this matrix, we may foretell the VM's behavior in the future. Using lasso regression and ridge regression, the CRMRCP method predicts a range of future CPU utilization levels based on historical values for VMs hosted on edge servers. The Markov models then use the expected future CPU usage values to label the condition of the servers as either overloaded, underloaded, or normally loaded. VMs that are projected to generate resource contention are transferred to other edge servers to avoid overloading the target edge servers. The number of VM relocations is analyzed, and comparisons are made between the DMRCP approach and the first-generation models. CRMRCP and the second-order Markov model are compared with regard to the number of VM migrations. In total, the results show that the DMRCP migrates by 52.9% compared to the first-order Markov model and by 21.1%. When ridge regression is cascaded with the second-order Markov model in CRMRCP, the sum of VM migrations is abridged by 81.8%.

Within the scope of algorithmic human resource management, Rodgers et al. [20] introduce the throughput model, which characterizes the decision-making processes of individuals. This model illustrates the impact of values, beliefs, and knowledge on decision-making and highlights the ways in which various approaches can be bolstered by the use of certain ethical decision-making algorithmic paths. This study draws on a wide range of theoretical perspectives, including those of AI-augmented human resource management (HRM(AI)) and HRM(AI) assimilation processes, AI-mediated social interchange, and the judgment and decision literature, to address questions about the impact and acceptance of AI integration in HRM. In this article, we propose the underexplored but crucial role of algorithmic ethical perspectives in the selection of HRM strategies, highlighting their utility in the adoption of AI for better HRM outcomes in terms of understandability and accountability.

To reduce the impact of network latency in asymmetric IoV settings, Cui et al. [21] devised a cloud-edge cooperative content-delivery strategy that makes the most of available computing, caching, and communication resources. This is a brief summary of the joint allocation problem of diverse resources, with a focus on minimizing delay according to the principles of queuing theory. The next step is for each network node to employ a novel deep reinforcement learning algorithm, which uses routing. Thorough simulations demonstrate the suggested technique converges quickly under a variety of circumstances and has lower network latency than existing alternatives in the cloud-edge scheme.

Tusa and Clayman [22] propose end-to-end slices as the basis for the thorough orchestration of compute incomes, network resources, and design that seeks to minimize the silo-effect. An operational system is offered as the means by which slices can be constructed dynamically across many domains, based on the graph-based model that is used to formally express the concept of the end-to-end slice. They are immutable entities that can be independently accessed and used to abstract resources from different providers that are then exchanged in a marketplace, with the bare-metal cloud allocation of compute slices communicating with one another via the connectivity of network slices. Using a real

testbed, we show that the end-to-end slices have the following three characteristics:

The HUNTER method, proposed by Tuli et al. [23], is a resource management strategy for environmentally friendly cloud computing. Energy, thermal, and cooling models are all taken into account in the projected model, which formulates the optimization of data center energy efficiency as a multi-objective scheduling issue. The GGCN is used as a proxy model in HUNTER to provide optimal scheduling decisions based on an approximation of the QoS for a given system state. HUNTER beats state-of-the-art baselines in terms of the energy-ingesting framework.

3. Proposed system

3.1 *System model*

Since the projected method is implemented in a cloud data center, we introduce and discuss all of its essential parts. We take into account aspects of the entire cloud service provisioning stack.

1. First, the software as a service (SaaS) layer processes incoming user workloads (whether batch-style or interactive) and sends them on to workload management, as described in Section 4. To begin with, the workload manager keeps the queue of tasks in a specified order depending on their priorities, taking into account QoS requirements like deadlines and budget limits.
2. Platform as a service (PaaS): This layer introduces a controller to manage the system's various operations. The controller optimizes the use of provided cloud resources with an eye towards improving the availability, efficiency, and sustainability of cloud data centers. In addition, a controller (middleware) is made up of the following five sub-modules, each of which performs a specific function: cooling manager, energy manager, fault manager, VM/resource manager, and workload manager.
3. The workload manager receives user workloads from the application imitations and passes that information on to the next module, the VM/resource manager, for use in resource provisioning and scheduling.
4. When a task has certain QoS criteria, the VM/resource manager will decide which virtual or physical machines to use to carry out the work.
5. A fault manager is able to detect and fix problems with minimal impact on system performance. In our study, we took into account three different kinds of errors: those that occur during the creation of VMs, those that occur on the host computer (such as when the processing elements fail or when memory runs out), and those that occur on a higher level, such as cloudlet failures due to networking issues. The data center manager functions as a middleman, coordinating the activities of specialized modules like the manager.

Information about cloud resources like VMs and container data stores is stored in the infrastructure as a service (IaaS) layer. The virtualization layer also permits workload balancing by migrating VMs. A proactive temperature-aware scheduler measures, monitors, and controls the temperature fluctuations of many VMs running on different cores. The power for the various parts of cloud data centers is distributed and managed by a power management unit. By saving the current states of VMs, DRAM serves as a check-pointing mechanism. Thermal sensors track the temperature and report its reading to the thermal profiling and monitoring module, which then examines the temperature fluctuations in cloud data centers. If the temperature in the data center rises over a predetermined limit, the system will trigger a thermal alert, at which point the heat controller will take the necessary corrective action. In the event of a power outage at the utility, an uninterruptible power supply is used to keep the lights on. The cloud data center's temperature is regulated by the district heating management system, which employs a water economizer, an outside air economizer, and a chiller plant. A manager of energy oversees both renewable and conventional power plants. More attention is paid to solar and wind power in sustainable cloud data centers [24]. The electricity generated from both renewable and non-renewable sources is controlled by an automatic transfer switch. In addition, all of the appliances and cooling components in cloud data centers receive their power from a power distribution unit.

3.2 *Workload model and problem formulation*

We follow the standard practice of viewing scheduling decision generation as a discrete-time control problem. We label these sections of the timeline with their respective durations, where the t-th section is referred to as I_t (with I commencing at 0). We label the group of cloud servers "H" and assume a constant number of hosts. Jobs J_t represent the

work to be done, and each job $j_i j_t$ is made up of several smaller tasks, denoted by t_0, t_1, etc. QoS metrics are determined on a job-by-job basis rather than on a task-by-task basis, although there are no priority limits between tasks that are part of the same job. Therefore, it is crucial to take job dependencies into account while making schedules. N_t represents the total number of newly generated jobs over the time period It, and A_t signifies the total sum of currently running jobs. If a job has at least one task running on the cloud, the job is deemed active. Any tasks from a job jN_t that cannot be immediately assigned to a cloud node are placed in a wait queue W_t. Metrics such as retort time and SLA breaches can be calculated for all new jobs that are not currently running or in the waiting queue.

Within the context of a time-constrained experiment, we focus on the challenge of optimizing the sum of the QoS objective scores for all time periods. We consider an entire set of n intervals, and we refer to the QoS score for interval I_t as O_t. For simplicity, we will refer to it as U_t. The next step involves making a scheduling prediction using U_t to arrive at a S_t value. Feasible tasks are those that can be accomplished in the time frame given by the parameters N_t, W_t, and A_t. With this, the issue can be stated as:

$$\underset{S_t}{maximize} \sum_{t=0}^{n} O_t$$

$$\text{Subject to } \forall t, S_t : P_t \cup Q_t \rightarrow H,$$

$$\forall t, P_t = \text{ set of feasible tasks in } Nt \cup Wt \cup At$$

$$\forall t, Q_t = \text{set of lively tasks in the scheme} \tag{1}$$

In the following discussion, we ignore the t subscript and exclusively use these symbols in reference to a single interval.

3.3 *Sustainability models*

In this work, we have developed separate energy, thermal, and cooling models to help separate these aspects of sustainable resource management [25].

3.3.1 *Energy model*

The goal of this strategy is to centralize all variables affecting energy consumption [26], from computing resources to cooling hardware. One can determine a cloud data center's total energy via

$$E_{Total} = E_{Computing} + E_{Cooling} \tag{2}$$

Hosts are the building blocks of a computing system, and all of its parts — the CPU, random access memory, discs, networks, and peripherals — have their own power needs. Hence, we may define $E_{Computing}$ as

$$E_{Computing} = E_{Processor} + E_{Storage} + E_{Memory} + E_{Network} + E_{Extra} \tag{3}$$

Processor. Here, $E_{Processor}$ signifies the processor's energy ingesting, which is intended by the addition of the idle and dynamic ingesting of all cores. Thus,

$$E_{Processor} = \sum_{r=1}^{cores} E_{dynamic}^{r} + E_{idle}^{r} \tag{4}$$

where $E_{dynamic}^{r}$ and E_{idle}^{r} are the power used while the r-th core is active and idle, respectively. Within this context, $E_{dynamic}$ is determined by

$$E_{dynamic} = \frac{E_{dynamic}^{linear} + E_{dynamic}^{non-linear}}{2}. \tag{5}$$

$E_{dynamic}^{linear}$ is intended as

$$E_{dynamic}^{linear} = CV^2 f, \tag{6}$$

where C is the CPU's capacitor, f the CPU's clock incidence, and V the CPU's voltage. To determine $E_{dynamic}^{non-linear}$, we use

$$E_{dynamic}^{non-linear}(h_j) = \mu_1.U_j + \mu_2.U_j^2 \tag{7}$$

where μ_1 and μ_2 are non-linear perfect limits, and U_j is the CPU utilization of host h_j.

Data storage energy usage is denoted by the variable $E_{Storage}$. A device's power consumption is determined by the number of times it reads and writes data.

$$E_{Storage} = E_{ReadOperation} + E_{WriteOperation} + E_{Idle} \tag{8}$$

E_{Memory} signifies the energy cache memory (SRAM), which is intended using

$$E_{Memory} = E_{SRAM} + E_{DRAM} \tag{9}$$

Network. $E_{Network}$ signifies the energy ingesting of networking apparatus, such as routers, switches and gateways, local area network (LAN) cards, etc., and is intended as

$$E_{Network} = E_{Router} + E_{Switches} + E_{Gateways} + E_{LAN\ cards} \tag{10}$$

Peripherals. E_{Extra} signifies the energy ingesting of other parts, with the current adaptation loss and others and is intended as

$$E_{\text{Extra}} = E_{motherboard} + \sum_{f \in F} E_{connector}^{f} \tag{11}$$

where $E_{motherboard}$ is energy-expended by motherboard(s), and $\sum_{f \in F} E_{connector}^{f}$ is energy-consumed by a connector running port incidences that is denoted by F.

3.3.2 *Cooling model*

In the cooling model, $E_{Cooling}$ means the amount of power required by cooling systems to keep a cloud data center at a constant.

$$E_{Cooling} = E_{AC} + E_{Compressor} + E_{Fan} + E_{Pump} \tag{12}$$

where E_{AC} is the power needed to run the air conditioner in the cloud data center, E_C is the power needed to run the compressor, E_F is the power needed to run the fans on the radiators, and E_P is the power needed to run the pump in the system.

3.3.3 *Thermal model*

For the CPU temperature (TCPU) estimation for each host (T_{cu}), we employ the computer room air conditioning (CRAC) model and the time-constant. Thus,

$$T_{cu} = PR + Temp_{inlet} + T_{initial} * e^{-RC} \tag{13}$$

where CRAC model (T_{cu}) is used to determine TCPU, and P is the host's dynamic power. Initial TCPU, denoted by T_{inital}, is set to the data center's ambient temperature.

4. Reserve provisioning and scheduling

Planning the execution of workloads with the given resources while keeping cloud services reliable and sustainable is a formidable challenge. Finding the optimal resource-workload pair based on user QoS criteria complicates cloud resource scheduling [27]. The issue can be stated as follows: it is desired to map a collection of autonomous cloud workloads to a collection of heterogeneous and mutable resources $(r_1, r_2, r_3, \ldots, r_n)$. $R = \{r_k \mid 1 \leq k \leq n\}$ is the set of available resources for a continuous problem, where n is the total number of available resources. $W = \{w_i \mid 1 \leq i \leq m\}$ is the collection of cloud workloads and m is the total number of cloud workloads. The system that identifies the best resources to use in order to execute user workloads is depicted in Subsection 5.2. These are the actions that must be taken for the proposed model to function: Workloads are first analyzed for their individual characteristics and QoS needs, then grouped together according to their shared needs, and finally, cloud resources are provided and scheduled for execution depending on the groupings.

Workload types and associated QoS requirements [27] are listed in Table 1 for this study.

Table 1. Cloud workloads and their key QoS supplies

QoS requirements	Workload name
Persistence and reliability	Backup and storage facilities
Correctness, high availability, and security	Endeavour software
High availability and security	Central financial services
Usability, internet accessibility, high availability, and safety	Online transaction processing
Variable computing load	E-commerce
Visibility, data backup, and network bandwidth	Graphics oriented
Computing capacity	Technological computing
High availability and reliable storage	Websites
Security and network bandwidth	Productivity applications
Usability, serviceability, and high availability	Critical internet applications
Portability and high availability	Mobile computing services
Testing time, suppleness, and user self-service degree	Software/project development and testing
SLA violation rate, resource utilization, cost, and period	Performance testing

4.1 *Resource provisioning*

Q-aware resource provisioning allocates computing resources for clustered workloads according to the needs of those workloads in each cluster. Workloads are submitted to the resource scheduler after the resources have been provisioned. After the workload is ready, the resource scheduler will ask for it to be submitted in exchange for the provided resources. Finally, the cloud user receives a response from the resource scheduler that includes the relevant resource data [28].

4.2 *Cat-based swarm optimization in resource scheduling algorithm*

Using a proposed scheduling technique, we aim to reduce cloud infrastructure's power usage while increasing reliability. Consolidating VMs onto fewer lively servers reduces system energy consumption, but server failure might impact numerous VMs and decrease system reliability; therefore, achieving these two goals simultaneously is often seen as a trade-off. Increasing the number of VM copies, on the other hand, maximizes system stability but comes with higher energy expenditures due to more computation being done and more servers being active. To mitigate this effect, providers of cloud services must strike a balance between energy efficiency and uptime. In particular, while energy management practices based on dynamic voltage and frequency scaling can reduce energy consumption, doing so comes with drawbacks in the form of increased response time and service delay as a result. Frequently powering on and off servers also degrades the system's dependability. Storage devices, RAM, and other server components are less reliable when subjected to power modulation. For this reason, innovative approaches to resource management are required to cut down on energy use with minimal effect on the availability of cloud services.

4.2.1 *Cloud service orchestrator (CSO) algorithm*

A domestic cat may lack the necessary hunting skills for survival, but it has an innate curiosity towards moving objects. Cats spend a lot of time napping, yet they are very alert while they are awake. They maintain vigilance in order to avoid threats. CSO can be thought of as having two distinct modes, seeking and tracing, that correspond to these two distinct behaviors. The algorithm divides the cats into seeking and tracing groups using a mixture ratio. While cats are napping, the situation is modeled using the seeking mode. There are four parameters that constitute the seeking mode: the seeking memory pool (SMP), the seeking ranger of selected dimensions (SRD), the counts of dimensions to change cloud data center, and the self-position considering (SPC).

Each cat's SMP is the size of its searching memory. Each cat will arbitrarily select a new memory location using a roulette wheel. To prevent values from shifting outside of an allowed range, SRD is applied to the selected dimension. The number of mutable dimensions is calculated using cloud data center. The SPC variable is a Boolean one. SPC is true (1) if and only if the cat is the healthiest individual (0). The SPC is one (1) and j = SMP, but just one copy of a new candidate if the SPC is zero (0).

Cats typically employ the tracing mode to follow a moving object or prey. Every tracking cat adjusted its position in response to the changing speed. The optimal position is used to update the cat's velocity in the CSO method. The CSO algorithm's operation is broken down into the following eight steps:

1. Make early N cats space.
2. Assess the fitness value and save the best site as *xbest*.
3. Gulf the cats into mode.
4. If the cat is in the seeking state, we can produce a new-fangled set of possible locations using equation (14), and then use the roulette wheel to select one of them as the new starting point.

$$x_{j,d}^{'} = x_{j,d} \pm SRD * r * x_{j,d} \tag{14}$$

where $x_{j,d}$ and $x_{j,d}^{'}$ are the current and new values of dimension d, and r is a random number.

Equation (15) is used to update the cat's velocity while in tracing mode, and equation (16) is used to update the cat's position (16)

$$v_{k,d}(t+1) = v_{k,d}(t) + c_1 r_1 \left(x_{best,d}(t) - x_{k,d}(t) \right) \tag{15}$$

$$x_{k,d}(t+1) = x_{k,d}(t) + v_{k,d}(t+1) \tag{16}$$

$$d = 1, 2, \ldots, D$$

where c_1 is acceleration coefficient; r_1 is a random quantity; $v_{k,d}(t), v_{k,d}(t+1)$, respectively, are velocity; and $x_{k,d}(t)$, $x_{k,d}(t+1)$ are current and novel position correspondingly.

6. Merge the cats from seeking and tracing style.
7. Appraise the fitness value and update the finest position.
8. Check the finish principle. If the standard is reached, then the procedure is stopped; else, back to step 3.

5. Results and discussion

In this section, we evaluate the experimental results of our proposed model. In this experimentation, we used a Python tool. In the below section, we describe the evaluation metrics of our proposed model.

5.1 *Evaluation setup*

We have used the COSCO framework [29] and the CloudSim toolkit [30] to conduct extensive testing in both realistic and simulated cloud settings, validating the effectiveness of our suggested strategy. We execute our trials for 100 scheduling intervals to get QoS findings, and the scheduling interval size remains constant at five minutes or 300 seconds. We need to average across five iterations for statistical significance. The first is a traditional data center configuration, and it makes use of 10 Azure VMs in a distributed cloud architecture.

- **Private cloud.** A total of six Azure machines: four B2s (dual-core CPU, 4 GB RAM) and two B4ms (quad-core with 16 GB RAM). They were created at the Azure facility in London, UK.
- **Public cloud.** Two 16 GB RAM and two B8ms (eight-core with 128 GB RAM) Azure machines (octa-core with 32 GB RAM). All of their beginnings may be traced back to the Azure data center in Virginia.

We also ran tests on a mock platform with 50 hosts, which is five times as many as the real platform. The former makes it possible to conduct more precise tests of our method, while the latter facilitates broad-scale experiments. To calculate the energy used by data centers, we run tests against the Standard Performance Evaluation Corporation (SPEC) power benchmarks. Based on the literature, we set R and C in (13) to be 0.50 and 0.03, respectively.

5.2 *Workloads*

The DeFog benchmarking programs' variety and lack of stationary workloads [31] make them ideal for our in-person tests. Yolo, PocketSphinx, and Aeneas are only a few of DeFog's AI apps that rely heavily on computational resources. Tasks like speech recognition, text-to-speech synchronization, visual object detection, and natural language processing (NLP) are all part of these. Then, we package these workloads into Docker containers so they can run on our cloud servers. Initially, for each scheduling window, we generate a set of Poisson() jobs with a ratio of 1:2. All three programs, Yolo, PocketSphinx, and Aeneas, contribute an equal number of job samples. For each individual work, we split the incoming batch into three to five pieces and placed them in different containers.

In our synthetic environment, we model a large-scale execution using the widely used dynamic traces dataset [32]. More than a thousand hosts' performance data from a diverse cloud data center is included in the dataset. The Alibaba distributed data center is a prominent provider of services for doing business calculations and managing hosting for industrial applications, and it is from there that these traces have been acquired. Among its most frequent visitors are insurance companies (Aegon), credit card companies (ICS), and a large number of banks (ING). Credit-related financial computing applications (such as Algorithmics and Towers Watson) are also housed here. Moreover, there are two types of traces: Rnd and fastStorage. We use both traces in our studies to accommodate a wide variety of workloads. These records are time series models of the CPU, random access memory, disk, and network throughput. Jobs are generated using the Poisson(5) distribution; like the physical setup, they each consist of three to five tasks drawn at random from the Rnd and fastStorage categories. Prior research is used to determine the value of the parameter. The experimental examination of the makespan of the proposed model in comparison to existing methodologies is shown in Table 2 and Figure 1. We take into account and apply all existing models to our system model, and then we take an average of the outcomes.

Table 2. Makespan of the proposed model

Number of works	Makespan (time in seconds)				
	Arithmetic optimization algorithm (AOA)	Butterfly	Rat	Whale	Proposed CSO
1,000	434	428	421	415	395
2,000	917	921	836	816	800
3,000	1,302	1,293	1,264	1,251	1,229
4,000	1,688	1,686	1,635	1,619	1,602
5,000	1,979	2,000	1,994	1,891	1,875

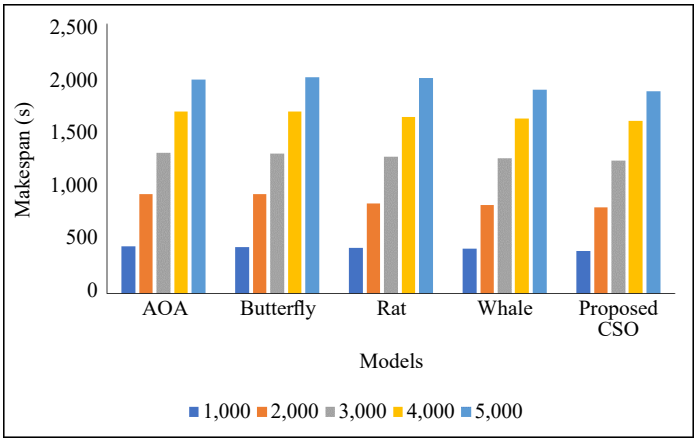

Figure 1. Makespan comparison

When the tasks were 2,000, the existing techniques achieved nearly 917 s, 921 s, 836 s, and 816 s, and the proposed model achieved 800 s. When the sum of tasks is high, the projected model achieves 1,602 s, whereas the existing models such as AOA, butterfly, rat, and whale achieve 1,688 s, 1,686 s, 1,635s, and 1,619 s. The reason for the better performance of the proposed model is that it will not easily fall into the local optimum. Table 3 and Figure 2 present the analysis of various models for energy consumption.

Table 3. The comparison of the proposed model using the energy consumption

Algorithms	Number of tasks					
	1,000	2,000	3,000	4,000	5,000	6,000
AOA	2,453	2,723	3,473	4,142	4,663	5,081
Butterfly	2,273	2,657	3,845	4,361	4,534	4,921
Rat	2,265	2,558	3,518	3,965	4,307	4,709
Whale	2,159	2,471	3,350	3,659	3,878	4,548
Proposed CSO	**2,055**	**2,206**	**3,032**	**3,435**	**3,726**	**3,976**

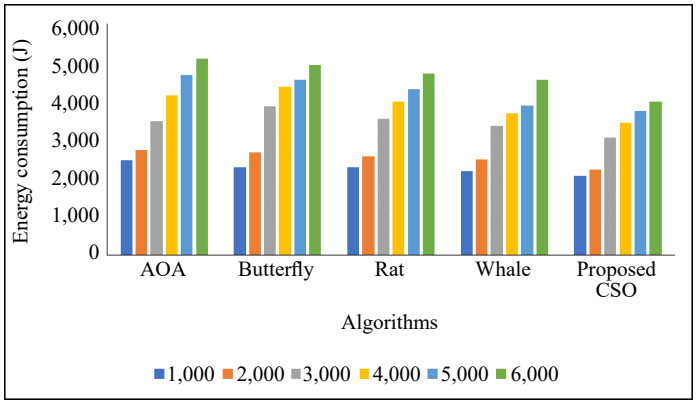

Figure 2. Comparison of the proposed model for energy consumption

When the task is 1,000, the AOA is 2,453 J, the butterfly achieved 2,273 J, the rat achieved 2,265 J, the whale achieved 2,159 J, and the proposed model achieved 2,055 J. The energy consumption is high when the tasks are high. For instance, the proposed model achieved 3,032 J, whereas the existing models achieved nearly 3,473 J to 3,350 J for the 3,000 tasks. The same models achieved nearly 4,663 J to 3,878 J, while the proposed model achieved 3,726 J for the 5,000 tasks. Table 4 and Figure 3 present the comparative analysis for execution overhead.

Table 4. The comparison of the proposed model using the execution overhead

Algorithms	Number of tasks					
	1,000	**2,000**	**3,000**	**4,000**	**5,000**	**6,000**
AOA	792	875	810	826	846	884
Butterfly	821	854	844	861	871	892
Rat	523	708	567	593	614	627
Whale	605	639	646	649	663	689
Proposed CSO	**482**	**503**	**537**	**542**	**551**	**561**

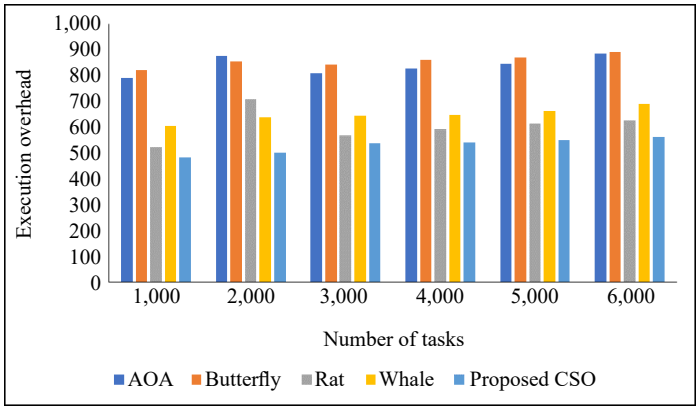

Figure 3. Execution overhead analysis

When the number of tasks is 1,000, the AOA achieved 792, the butterfly achieved 821, the rat achieved 523, the whale achieved 605, and the proposed model achieved 482. When the task is 3,000, the existing models such as AOA and butterfly achieved 810 and 844, the rat 567, the whale 646, and the proposed model 537. When the task is 6,000, AOA has 884, butterfly has 892, rat has 627, whale has 689, and the proposed model has 561.

6. Conclusions and future work

In this research, we advocated for cat-based swarm optimization for comprehensive resource management scheduling. Our scheduler makes it possible to use cloud servers in a way that minimizes energy consumption and thermal hotspots. This is made possible by CSO's novel use of cooling-specific energy and temperature models. Also, the suggested approach may be utilized to rapidly provide QoS estimations, saving a substantial amount of money that would otherwise be spent testing out different scheduling options. CSO makes use of a performance-to-power ratio as a heuristic to equitably distribute workloads among cloud hosts, thereby maximizing computational power while decreasing energy usage. Additionally, this algorithm permits rapid iteration through the scheduling of search space and decision-making. Numerous tests on both physical and artificial testbeds indicate that CSO performs better than baselines in various QoS measures. The suggested model is efficient in managing resources while handling dynamic workloads, as evidenced by the tiny values of the coefficient of variation of energy and temperature. The results of the studies show that the suggested model outperforms the currently available resource schedulers based on heuristic algorithms derived from AI (AOA, butterfly, rat, and whale). The scope of this task can be broadened by considering additional criteria, such as scalability, security, and reliability, as well as their impact on energy consumption. Domain-specific strategies for cooling, such as those used in industries that rely heavily on the Internet of Things and fog/edge computing like agriculture, healthcare, and smart homes, may be a focus of future research.

Conflict of interest

There is no conflict of interest in this study.

References

[1] Saxena S, Khan MZ, Singh R. Green computing: An era of energy saving computing of cloud resources. *International Journal of Mathematical Sciences and Computing*. 2021; 7(2): 42-48. Available from: https://doi.org/10.5815/ijmsc.2021.02.05.

[2] Qi S, Huang Z, Ji L. Sustainable development based on green GDP accounting and cloud computing: A case study of Zhejiang Province. *Scientific Programming*. 2021; 2021: 7953164. Available from: https://doi.org/10.1155/2021/7953164.

[3] Ahmad N. The structural modeling of significant factors for sustainable cloud migration. *International Journal of Intelligent Engineering and Systems*. 2021; 14(2): 1-10. Available from: https://doi.org/10.22266/ijies2021.0430.01.

[4] Oke AE, Kineber AF, Albukhari I, Othman I, Kingsley C. Assessment of cloud computing success factors for sustainable construction industry: The case of Nigeria. *Buildings*. 2021; 11(2): 36. Available from: https://doi.org/10.3390/buildings11020036.

[5] Oke AE, Kineber AF, Al-Bukhari I, Famakin I, Kingsley C. Exploring the benefits of cloud computing for sustainable construction in Nigeria. *Journal of Engineering, Design and Technology*. 2023; 21(4): 973-990. Available from: https://doi.org/10.1108/JEDT-04-2021-0189.

[6] Narayana KE, Jayashree K. Survey on cross virtual machine side channel attack detection and properties of cloud computing as sustainable material. *Materials Today: Proceedings*. 2021; 45: 6465-6470. Available from: https://doi.org/10.1016/j.matpr.2020.11.283.

[7] Mustapha UF, Alhassan A-W, Jiang D-N, Li G-L. Sustainable aquaculture development: A review on the roles of

cloud computing, internet of things and artificial intelligence (CIA). *Reviews in Aquaculture*. 2021; 13(4): 2076-2091. Available from: https://doi.org/10.1111/raq.12559.

[8] Zheng J, Wang Y. A hybrid multi-objective bat algorithm for solving cloud computing resource scheduling problems. *Sustainability*. 2021; 13(14): 7933. Available from: https://doi.org/10.3390/su13147933.

[9] Shah SQA, Khan FZ, Ahmad M. The impact and mitigation of ICMP based economic denial of sustainability attack in cloud computing environment using software defined network. *Computer Network*. 2021; 187: 107825. Available from: https://doi.org/10.1016/j.comnet.2021.107825.

[10] Oke AE, Kineber AF, Abdel-Tawab M, Abubakar AS, Albukhari I, Kingsley C. Barriers to the implementation of cloud computing for sustainable construction in a developing economy. *International Journal of Building Pathology and Adaptation*. 2021; 41(5): 988-1013. Available from: https://doi.org/10.1108/IJBPA-07-2021-0098.

[11] Forcén-Muñoz M, Pavón-Pulido N, López-Riquelme JA, Temnani-Rajjaf A, Berríos P, Morais R, et al. Irriman platform: Enhancing farming sustainability through cloud computing techniques for irrigation management. *Sensors*. 2021; 22(1): 228. Available from: https://doi.org/10.3390/s22010228.

[12] Ullah Z, Umer A, Zaree M, Ahmad J, Alanazi F, Amin NU, et al. Negotiation based combinatorial double auction mechanism in cloud computing. *Computers, Materials & Continua*. 2021; 69; 2123-2140. Available from: https://doi.org/10.32604/cmc.2021.015445.

[13] George SS, Pramila RS. A review of different techniques in cloud computing. *Materials Today: Proceedings*. 2021; 46: 8002-8008. Available from: https://doi.org/10.1016/j.matpr.2021.02.748.

[14] Rico-Bautista D, Guerrero CD, Collazos CA, Maestre-Gongora G, Sánchez-Velásquez MC, Medina-Cárdenas Y, et al. Smart university: Key factors for a cloud computing adoption model. In: Nagar AK, Jat DS, Marín-Raventós G, Mishra DK. (eds.) *Intelligent Sustainable Systems: Selected Papers of WorldS4 2021, Volume 2.* Singapore: Springer; 2021. p.85-93. Available from: https://doi.org/10.1007/978-981-16-6369-7_8.

[15] Singh G, Singh P. A taxonomy and survey on container migration techniques in cloud computing. In: Singh H, Singh PP, Garg P. (eds.) *Sustainable Development Through Engineering Innovations: Select Proceedings of SDEI 2020*. Singapore: Springer; 2021. p.419-429. Available from: https://doi.org/10.1007/978-981-15-9554-7_36.

[16] Hussain M, Wei LF, Lakhan A, Wali S, Ali S, Hussain A. Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. *Sustainable Computing: Informatics and Systems*. 2021; 30: 100517. Available from: https://doi.org/10.1016/j.suscom.2021.100517.

[17] Iftikhar S, Ahmad MMM, Tuli S, Chowdhury D, Xu M, Gill SS, et al. HunterPlus: AI based energy-efficient task scheduling for cloud-fog computing environments. *Internet of Things*. 2023; 21: 100667. Available from: https://doi.org/10.1016/j.iot.2022.100667.

[18] Karthikeyan R, Balamurugan V, Cyriac R, Sundaravadivazhagan B. COSCO2: AI-augmented evolutionary algorithm based workload prediction framework for sustainable cloud data centers. *Transactions on Emerging Telecommunications Technologies*. 2023; 34(1): e4652. Available from: https://doi.org/10.1002/ett.4652.

[19] Surya K, Rajam VMA. Novel approaches for resource management across edge servers. *International Journal of Networked and Distributed Computing.* 2023; 11: 20-30. Available from: https://doi.org/10.1007/s44227-022-00007-0.

[20] Rodgers W, Murray JM, Stefanidis A, Degbey WY, Tarba SY. An artificial intelligence algorithmic approach to ethical decision-making in human resource management processes. *Human Resource Management Review*. 2023; 33(1): 100925. Available from: https://doi.org/10.1016/j.hrmr.2022.100925.

[21] Cui T, Yang R, Fang C, Yu S. Deep reinforcement learning-based resource allocation for content distribution in IoT-edge-cloud computing environments. *Symmetry*. 2023; 15(1): 217. Available from: https://doi.org/10.3390/sym15010217.

[22] Tusa F, Clayman S. End-to-end slices to orchestrate resources and services in the cloud-to-edge continuum. *Future Generation Computer Systems*. 2023; 141: 473-488. Available from: https://doi.org/10.1016/j.future.2022.11.026.

[23] Tuli S, Gill SS, Xu M, Garraghan P, Bahsoon R, Dustdar S, et al. HUNTER: AI based holistic resource management for sustainable cloud computing. *Journal of Systems and Software*. 2022; 184: 111124. Available from: https://doi.org/10.1016/j.jss.2021.111124.

[24] Guitart J. Toward sustainable data centers: A comprehensive energy management strategy. *Computing*. 2017; 99(6): 597-615. Available from: https://doi.org/10.1007/s00607-016-0501-1.

[25] Gill SS, Garraghan P, Stankovski V, Casale G, Thulasiram RK, Ghosh SK, et al. Holistic resource management for sustainable and reliable cloud computing: An innovative solution to global challenge. *Journal of Systems and Software*. 2019; 155: 104-129. Available from: https://doi.org/10.1016/j.jss.2019.05.025.

[26] MirhoseiniNejad S, Moazamigoodarzi H, Badawy G, Down DG. Joint data center cooling and workload management: A thermal-aware approach. *Future Generation Computer Systems*. 2020; 104: 174-186. Available from: https://doi.org/10.1016/j.future.2019.10.040.

[27] Gill SS, Buyya R. Resource provisioning based scheduling framework for execution of heterogeneous and clustered workloads in clouds: From fundamental to autonomic offering. *Journal of Grid Computing*. 2019: 17: 385-417. Available from: https://doi.org/10.1007/s10723-017-9424-0.

[28] Singh S, Chana I. Q-aware: Quality of service based cloud resource provisioning. *Computers & Electrical Engineering*. 2015; 47: 138-160. Available from: https://doi.org/10.1016/j.compeleceng.2015.02.003.

[29] Tuli S, Poojara S, Srirama SN, Casale G, Jennings N. COSCO: Container orchestration using co-simulation and gradient based optimization for fog computing environments. *IEEE Transactions on Parallel and Distributed Systems*. 2022; 33: 101-116. Available from: https://doi.org/10.1109/TPDS.2021.3087349.

[30] Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*. 2011; 41(1): 23-50. Available from: https://doi.org/10.1002/spe.995.

[31] McChesney J, Wang N, Tanwer A, de Lara E, Varghese B. Defog: Fog computing benchmarks. In: *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing (SEC '19)*. New York, USA: Association for Computing Machinery; 2019. p.47-58. Available from: https://doi.org/10.1145/3318216.3363299.

[32] Shen S, Van Beek V, Iosup A. Statistical characterization of business-critical workloads hosted in cloud datacenters. In: *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. Shenzhen, China: IEEE; 2015. p.465-474. https://doi.org/10.1109/CCGrid.2015.60.