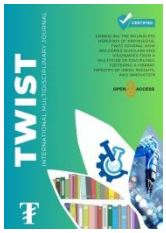
DOI https://doi.org/10.5281/zenodo.10049652#263

TWIST, 2024, Vol. 19, Issue 3, pp. 559-571

TWIST

Journal homepage: www.twistjournal.net



Optimized Resource Sharing in Mobile Cloud Computing for Reducing Service Latency

J. Rathika*
Assistant Professor, Department of Computing, Coimbatore Institute of Technology, Coimbatore, India
[**Corresponding author*]

Dr. M. Soranamageswari
Assistant Professor, Department of Information Technology, Government Arts College, Coimbatore, India

Abstract
Fog computing is a notion that merges the benefits of cloud and edge devices to furnish excellent services, diminish delay, offer mobility assistance, multi-user access, and other characteristics that maintain current computing systems. It is also identified as fogging or fog networking. This write-up introduces a numerical structure for dissimilar resource allocation via task-driven usefulness functions based on the combination of Mayfly and Reformed Multi-Objective Particle Swarm Optimization (RMPSO), referred to as MRMPSO. An idea with the organization of multi-goal function is brought and projected to prior to a fresh and innovative optimization set of rules for updating the rate of the fundamental mayfly set of rules. The RMPSO with function of the crossover series permits each and every mayfly adopting its non-public conduct to make it quicker with the organization information. In multi-level mayfly set of rules, speed is updated, based at the offspring's first-class function of the mayfly. At the identical time, it is brought with prominence to offer higher resource sharing characteristics. This framework includes cloud computing, mobile cloud computing (MCC), and mobile edge computing (MEC), with the aim of enhancing the QoS (quality of service) among terminal devices and the cloud. Taxonomy of MCC is proposed based on current research in the field, covering confidential challenges, data management, operational and service issues. The review identifies challenges and potential applications in cloud computing, with security, privacy, communication and application challenges being prominent in addition to the academic contributions. The article also explores potential applications of MCC, such as healthcare, smart city, and agriculture applications.

Keywords
Mobile cloud, Mayfly, PSO, Hybrid optimizer, Resource sharing, Service-oriented functions

INTRODUCTION

The latest developments in mobile devices and cloud computing platforms have led to the emergence of MCC, a novel approach to mobile services and applications that are expected to have a significant influence on our daily lives. Initially, MCC was primarily concerned with enhancing the ability of mobile nodes to compute and store data through delegating tasks to more robust cloud data centers [1].

The initial type is referred as agent-client-based mobile cloud architecture [2, 3]. It relies on a centralized data center to exclusively provide resources, such as CPU and storage, to mobile devices and carry out the required operations to execute a service. In this configuration, mobile devices solely use cloud resources; they make no service contributions [4]. The next type of architecture is known as a collaboration-based mobile cloud architecture, in which the central mobile devices and data centers collaborate by sharing their assets to provide services.

Such designs can differ greatly and are particularly potent because of the vast number of devices available to participate in the cloud. This concept is also known as fog computing [5], which broadens the cloud computing model to the network's edge. The cooperative-based mobile cloud is currently the most fascinating and noticeable research field of MCC. The latest advancements in the proliferation and variety of mobile devices have introduced numerous dissimilar assets into local networks, such as efficient processors, rapid Long Term Evolution (LTE) connections, voluminous storage, and various sensor data. These assets pertain to computational, communicative, storage, and informative

resources, respectively. Taking advantage of these different resources when possible permits cooperative cloud computing to function on advanced mobile devices, rendering it a resilient foundation for accommodating a diverse array of applications.

There are still more technical challenges that must be addressed with the aim of fully implementing the cooperative mobile cloud. This study centers on the primary issue of how to effectively coordinate the sharing of diverse resources among different nodes. Traditional methods of facilitating resource sharing typically focus on coordinating tasks, such as numerical computations or data downloads, without taking into account the specific services being provided [2, 3]. In such task-driven sharing, diverse resources are frequently evaluated using varying scales or units (e.g. bandwidth, power, latency etc.), and scheduling tasks are assigned to optimize specific metrics based on these differing scales or units.

A common goal is to create methods that allow for energy-efficient CO with low latency. For instance, [5] presents an optimized approach for a single-user MEC, where the energy consumption is minimized by jointly controlling CPU cycles and the decision to offload based on the estimation deadline necessity. In [6], a joint allocation of computation and radio resources is proposed for a multi-user MEC with respect to each deadline limitation while minimizing the total amount of mobile energy use. The best method in [6] is based on assumptions that might not hold true in real-world situations, such as a set number of mobiles, synchronous task arrival, and identical and deterministic computing deadlines.

Several recent studies attempt to address the boundaries by lessening the aforementioned assumptions. The text in [7] investigates an MEC scenario that is asynchronous, wherein mobile devices experience different times for input data arrival and computation deadlines. Meanwhile, [8] establishes multi-user scheduling for MEC with random user arrivals using reinforcement learning and the Markov decision process. [9] studies the probabilistic deadline constraint, which guarantees that the probability of task queue length exceeding the allowable limit is less than the tolerable violation probability. This constraint is crucial in mission-critical applications. The researchers approximate the constraint to a generalized Pareto distribution through the application of the extreme value theory. This approximation makes it possible to determine a dynamic offloading decision and resource allocation. Furthermore, [10] and [11] deal with computation uncertainty, where the process of computation is random instead of deterministic, and the type and amount of necessary resources are unknown at the start.

In [10], the method of real-time preloading is suggested to reduce the projected power usage when dealing with a stochastic computing procedure that is designed as a Markov chain. In [11], the issue of determining whether to offload is tackled without depending on any specific details about the intended applications, like the number of CPU cycles, the amount of data to be uploaded or downloaded, by utilizing a chance restraint programming tool.

The existing investigations discussed above concentrate on enhancing the offloading of a computation task that contains user-specific data (USD) from mobile devices to the edge. These studies assume that the service-specific data (SSD) is already pre-installed. For instance, in the case of augmented reality (AR), visual recognition and the object database models must be organized at the edge to improve the classification or object recognition outcomes in the image [12]. There is an assumption valid if the services that can operate at the edge are restricted and well-anticipated. Nonetheless, a wide a diverse range of computing services ought to be requested from the edge, encompassing both conventional light services and intensive services that utilize the latest advancements in machine learning [13, 14]. Installing SSDs in advance for all conceivable services is not feasible. A novel approach is therefore necessary for the acquisition of both (SSD) and (USD).

Sharing that is focused on task for this particular service reduces the amount of time it takes to process each task. Nevertheless, the navigation service possesses a unique characteristic whereby users cannot utilize the service until they have finished downloading task, even though the route calculation takes only a few hundred microseconds. In this scenario, it is not logical to optimize computational resources as doing so could possibly lead to a waste of resources. Instead, it would be more beneficial to optimize a utility function that is oriented towards the service, which can better encompass the advantages of such optimization.

This article presents a framework and structure for sharing heterogeneous resources based on task-oriented utility functions. As varying resources are often measured differently, a unified system is offered that maps all quantities to time resources. The overall contributions of our MRMPSO based task scheduler and resource sharing is being pointed out here.

- Our proposed resource sharing algorithm combines the Mayfly algorithm with the Reformed PSO (RMPSO) skill, referred to as MRMPSO.
- The mayfly algorithm minimizes the search space, while the RMPSO identifies the improved response.
- The RMPSO algorithm selects the best global (gbest) a particle with a close point-to-line distance.
- This method elects ‘gbest’ particle candidates using the minimum distance from a point-to-line, while this scheme is used to recognize the improved response, in order to reduce the search space.
- The proposed hybrid optimized algorithm achieves an efficient average load and improves crucial factors like optimum resource usage and job response time.
- The resulting mobile cloud has the potential to be a robust platform for mobile services and cloud applications.
- Our simulation results demonstrate that the MRMPSO model performs effectively compared to other methods.

Following is a projected outline of the upcoming sections. In Section 2, pertinent studies are covered. The proposed methodology of applied algorithm is outlined in Section 3; the modified mayfly and reformed form of PSO sketched in Section 4 and they are assessed in Section 5. At last, the conclusions with the future possibility are outlined in Section 6.

RELATED APPROACHES

Cloud computing (CC) is referred to as a shared virtual distributed computing system among its clients across a wide geographical area using the internet. Multiple cloudlets can request and utilize the resources simultaneously. The centralized resource can be accessed by clients from anywhere and at any time through the internet.

The allocation and management of cloudlets on virtual machines (VMs) in an efficient manner to reduce computational costs is referred to as load management. This technique is employed to minimize the time of transfer, response, waiting, and execution time, as well as operational expenses.

Various approaches have been explored in existing research to optimize allocation and task scheduling on MCC with altered objectives. In order to minimize energy consumption, it is crucial for mobile apps to provide complex high-performance features. The energy-intensive components of mobile devices include I/O processing and network communications [15]. To address this issue, offloading resource-intensive tasks to the cloud is an efficient way to reduce energy usage. By augmenting mobile computation, intensive computations can be delivered to mobile users, which help to conserve device energy and extend battery life. Literature has proposed several energy-based models, such as remote resource cloning for mobile devices [16], code offloading and migration [17], network profiling & its applications [18], decomposition, reusability & their applications [19, 20].

The handling of extensive data files can cause a direct drain on the device's energy on mobile devices. Conversely, the amount of device idling time, overall cost, and processing time may all raise as a result of massive data file transfers across mobile networks. The literature indicates a heightened focus on resource scalability and elasticity. Optimization models for resource dependent MCC are concentrated on cloud resource scaling and virtual machine parallelism [21], segmentation of workflow-aware execution partitioning, hybrid MCC deployment, and elastic applications [22]. The importance of consuming intelligent and scalable context-aware systems [23] is emphasized for mobile applications that can manage the dynamic cloud mobile environment.

Nan et al. [24] investigated the difficulties of data-intensive mobile apps for the purpose of optimizing costs. They emphasized the importance of cost optimization to enhance user quality of service (QoS). To achieve this, an efficient cost optimization model is needed that minimizes monetary costs while enhancing customer QoS. A middleware for MCC with three tiers, which presents programmers with multiple computing choices, is introduced depending on the cost model of the application and the decision maker for offloading [25]. Despite taking into account the size of task data to create a streamlined execution plan for application tasks, this model fails to accurately depict the scheduling scenario of data-intensive applications on MCC due to the small size of the data.

On the other hand, energy-based MCC solutions mainly focus on optimizing device energy by migrating application code complexity. However, they do not take into account context variables like the dimensions of input data, network capacity, and associated data transfer expenses must be considered. Enhancing mobile network performance leads to better application response time and energy utilization optimization [26]. Application response time and availability can be enhanced by outsourcing tasks to nearby cloudlets [27]. Determining whether an application should be run on a local or remote basis is a challenging job that necessitates ongoing monitoring of network conditions and application profiling [19].

While assigning various workloads, load allocation can be done through either dynamic or static scheduling techniques [28]. Bio-inspired algorithms are utilized in dynamic techniques to schedule the workloads. CC employs swarm-based algorithms to assign loads to consecutive objects based on the velocity and location of particles. Different load balancing techniques have their own advantages and disadvantages. The round-robin algorithm is integrated into the scheduling model which works on the basis of meta-heuristic models. After several iterations and fitness values, the roulette selection model is determined based on random scheduling. Static algorithms are not categorized under meta-heuristic techniques used for managing the load in the cloud. When compared to rudimentary approaches like FCFS (First Come First Serve) and Round Robin scheduling, the operational cost incurred by this technique is low.

An algorithm named PSO has been introduced as a meta-heuristic approach [29]. This technique employs a group of particles that mimic the behavior of birds from one source of transition to other. The optimal location in the search space is determined based on the velocity of the preceding particle. In [30], various scheduling approaches such as GA, Min-Min, SA, Tabu Search, Max-Min, etc. are designated using static and alternate models.

This study focuses on the correlation between the Ant Colony Optimization (ACO) techniques. The discussion revolves around the role of ants in finding food, as the name suggests. It proposes the implementation of a PSO algorithm to allocate resources in the cloud on VM. The operation of this optimizer [31] has presented that leads to enhanced solutions.

In [32] literature, a PSO scheme based on constraints was utilized to distribute tasks across sequential nodes. The PSO algorithm has outlined in [33] collected works when implemented on a CC platform. The fitness function evaluates the most suitable particle, and its velocity is determined by the particle's pbest and the swarm's gbest. The MCC can include the latest swarm approach, mayfly, which boasts unique optimization capabilities. The position of the crossover

collection, along with the distribution function, allows each mayfly to adopt its own behavior and speed up with group information [34-36]. It lowers the overall calculation cost in comparison to earlier works and related techniques.
The primary shared problem among the aforementioned studies is the disregard for the magnitude of data and intricacy of MCC application in MCC structures. To address this limitation, a contemporary optimization method is proposed for running data-heavy MCC applications on the hybrid-optimized platform.

METHODOLOGY USED

The plan is to introduce a framework for optimizing MCC that is focused on data. This will be done in a hybrid-MCC environment that utilizes three different types of resource allocation: cloudlet, public cloud, and mobile device. Cloudlet is responsible for providing computation services to mobile clients that require sensitive requests. Accessible resources got through Wi-Fi or cellular networks. The public cloud affords resources that are scalable and powerful. For mobile devices, it is assumed that they are operating within the restrictions of energy and wireless interfaces, they can carry out local computation and storage.

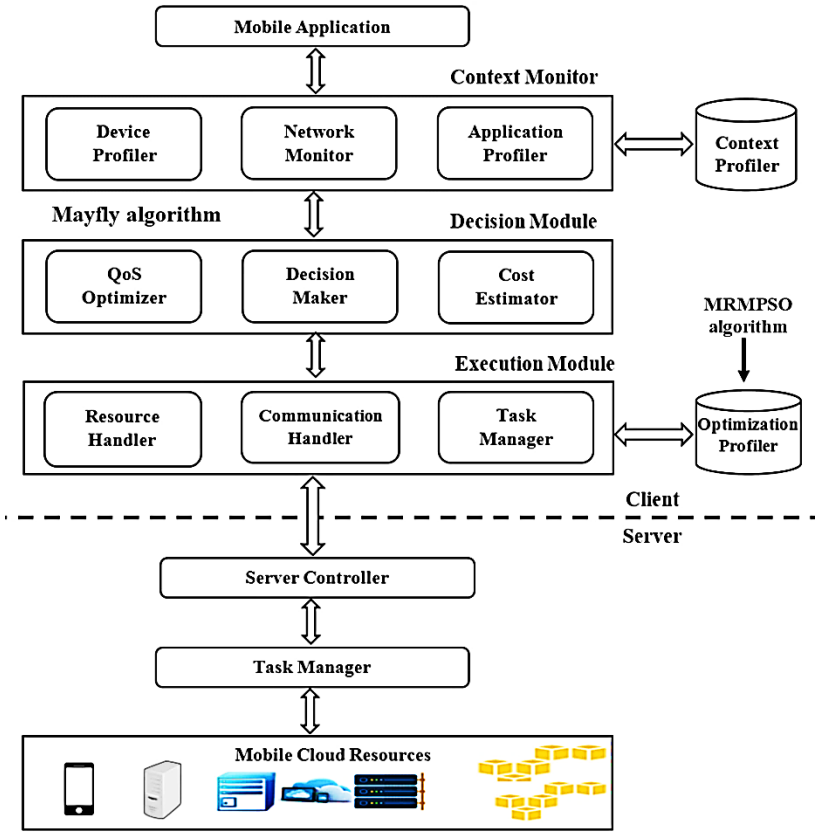


Fig 1. MRMPSO blocks with mobile cloud data

The suggested optimization framework for offloading comprises of modules that offer context monitoring, decision-making, and application execution services. The framework is depicted in Fig. 1.

The task of the context monitoring module is to create a profile of context parameters during runtime and assist the decision maker in making decisions based on energy and monetary cost estimations. The framework provides three profilers, namely, the device profiler for energy consumption, the network monitor for mobile network information availability, and the application profiler for recording heuristic data about application execution, taking into account network bandwidth context information.

The module responsible for decision-making is a QoS enhancer that endeavors to identify the optimal plan for executing applications within a solution space where the cost estimator evaluates each solution. The determination is founded on the user's QoS and estimated time for execution, along with the overall energy consumption and financial expense of the device.

The execution module is in charge of executing the application execution plan received from the decision maker. It comprises three primary components: the resource manager that handles resource access for storage and computation, the communication manager that oversees the communication networks between resources, and the task manager that executes multiple application tasks.

Scheduling tasks on virtual machines with limited resources is a challenging problem that requires optimizing an objective function while adhering to specific constraints. In the context of cloud computing, the effectiveness of task scheduling is evaluated using various system performance metrics. These optimization criteria can be broadly categorized into two types: those that cater to the demands of cloud users and those that cater to the demands of cloud service providers. The former includes user demand criteria like makespan (MS) and total execution time (TET), while the latter includes provider demand criteria like degree of imbalance (DI) and energy consumption. In single objective task

scheduling, only one criterion is taken into account, whereas in multi-objective criteria, two or more parameters are considered as objectives [36, 37].

A blend of firefly and enhanced particle swarm optimization (IPSO) algorithm are involved to achieve load balancing and task scheduling in a cloud computing setup. The IPSO algorithm's stochastic operators render it susceptible to the initial population. The algorithm's convergence is notably influenced by the improper selection of the initial population. To counter this, we have employed the firefly algorithm to initialize the IPSO [38, 40].

The combination of two optimizers of mayfly and reformed multi-objective PSO is utilized effectively to enhance the essential measures like proper resource usage and response time of the tasks. Initially, the mayfly algorithm is used in the decision module for making the best decision and this is further combined with the new model of PSO in the execution module for the better accomplishment of complete progress. Thus, the MRMPSO is utilized for the multi-purpose progress in resource sharing, task scheduling, and decision making.

PROPOSED MODEL

The hybrid optimizer in the proposed module, namely PSO and mayfly are combined with some modifications in their progress to assess the best optimum value for task scheduling and decision making in multiple levels. These optimization processes are described as follows.

Mayfly concept

A new and innovative optimization technique is presented in the form of the Mayfly optimization method [34-36]. Drawing inspiration from the mating and fighting behaviors of mayflies, such as their wedding dance, random movement, group gathering, mutation, and mayfly crossover, this algorithm was developed. The standard Mayfly algorithm operates in the following manner.

- Fix the speed settings and initialise the male and female mayfly populations.
- Calculate the fitness value and sort the outcomes to obtain sbesti and cbest.
- Adjust the locations of the male and female mayflies in turn before mating.
- Once fitness has been determined, update sbest and cbest.
- Verify whether the stop condition has been met; if so, exit and output the result; if not, repeat steps 3 through 5.

Both mayfly collections are initially created using random representations of the populations of male and female mayflies, respectively [35]. Each mayfly is randomly placed as a prospective candidate solution represented by the vector p = ($p_1,\ldots,p_d$), with dimension d, and after that, its performance is evaluated using ***f*(x)**, an objective function that has been predetermined. Each mayfly's flying direction is a dynamic combination of social flying and personal experiences. A mayfly's velocity, mv= ($mv_1,\ldots,mv_d$), is shown as a shift in position.

Male mayflies

Male mayflies congregate in clusters, and their arrangement is fine-tuned through self-experience and observation of their peers. If we consider that is the present location of mayfly i in the exploration domain during time step t, the updated position is the sum of the velocity at iteration $(t+1)^{th}$ and the position at iteration t^{th}, and its positional formula is altered as,

$$p_i^{t+1} = p_i^t + mv_i^{t+1} \qquad (1)$$

in which p_i^t denotes the mayfly's present location in the searching space i at time step t, mv_i^{t+1} velocity in the similar location.

$$mv_{ij}^{t+1} = mv_{ij}^t + \sigma_1 \, e^{-\gamma r_p^2}\left(sbest_{ij} - p_{ij}^t\right) + \sigma_2 \, e^{-\gamma r_g^2}\left(cbest_j - p_{ij}^t\right) \qquad (2)$$

where v_{ij}^t is the velocity (i) at time step t of mayfly, (ij = 1,...,n) in dimension j, two constants of positive attraction σ_1 and σ_2. s$best_i$ is the mayfly was in its best (i) ever position. Additionally, in eq. (1), the visibility coefficient (γ) fixed is used. This is done to reduce a mayfly's ability to reflect light on other flies.

Female mayflies

With care for fitness function, the best male mayfly should be drawn to the best female. Each mayfly can specifically go in the direction of what it perceives to be in the optimal position. Any mayfly in the swarm right now needs to get in the ideal posture (c*best*).

The best guy ought to be drawn to the best female while taking fitness into account. Chiefly, each mayfly has the ability to move in the direction of its subjective/individual ideal position (s*best*). Currently, any mayfly in the swarm must take the best location (c*best*).

$$mv_{ij}^{t+1} = \begin{cases} mv_{ij}^t + \sigma_2 \, e^{-\gamma r_{mf}^2}\left(p_{ij}^t - q_{ij}^t\right), & if\ f(q_i) > f(p_i) \\ mv_{ij}^t + f_l * r\ , & if\ f(q_i) \le f(p_i) \end{cases} \qquad (3)$$

q_{ij}^t denotes the position (i) of female fly at step (t) time in j dimension.

Mating progress

The crossover factors represent the mating behavior of both mayflies as follows. Following are both of the children that are created with respect to mating,

$$offspr1 = \mathrm{M} * male + (1 — \mathrm{M}) * female \quad (4)$$
$$offspr2 = \mathrm{M} * female + (1 — \mathrm{M}) * male \quad (5)$$

In which, the parents of the male (father) parent and mother are created from female and male, respectively, and M is a random number within a predetermined range (as in Eq. (4 & 5)). The beginning velocities of the offspring are set to zero. *M* stands for the Gauss distribution with any random statistic. On the other hand, there is initially a tuning problem with the mayfly optimisation. To increase this progress efficiency, a weighted parameter is also added. Finally, MO is proposed here to deal with the multi-objective problems.

Reformed mayfly progress

It is possible to compute the peculiar best position ($sbest_{ij}$), following at step time *t+1*, by taking equation (1) with minimization problems into consideration as,

$$pbest_i = \begin{cases} p_i^{t+1}, & if\ f(p_i^{t+1}) > f(sbest_i) \\ same, & otherwise \end{cases} \quad (6)$$

where f:$\mathbb{R}n \rightarrow \mathbb{R}$ represents the objective criteria for assessing the quality of the answer. The complete group of male mayflies (in swarm) can also be defined as M, the global (topmost) position $gbest$ at time step *t* which is designated as,

$$gbest\ \epsilon\{sbest_1 sbest_2, \dots, sbest_M \mid f(cbest)\}$$
$$= \min\{f(sbest_1), f(sbest_2), \dots, f(sbest_M)\} \quad (7)$$

To improve the M parameter, we have implemented the Modified Gaussian (MG) model. By integrating the breeding outcomes with this MG model, it can be quickly resolved by adjusting specific aspects of the breeding position incorporated in the optimization algorithm. The result in the prompt convergence of the algorithm is preserving the attributes of the traditional process.

$$RG_\sigma(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}\left(\frac{x-c}{\sigma}\right)^2} \quad (8)$$

In which $G_\sigma(y), \sigma, \theta$ and c are denoted as Gaussian function of y variable, variance, orientation that must be used to rotate the function and expected value respectively. The ultimate result attained by combining the RG with the last generation is presented in the equation below. To prevent premature convergence, wherein the local optimal value is considered superior to the global value, the selected mayfly progeny undergoes mutation by adding the distributed random number, RG. The formula for the mutation of mayfly progeny is as follows.

$$offspr_n = offspr_n + \emptyset\{\mathrm{RG}\} \quad (9)$$

$$\text{where, } \emptyset = \begin{cases} 1 & if\ (m_n^{i+1} - m_n^i = 0) \\ 0 & else \end{cases} \quad (10)$$

In the aforementioned equations, the probability factor $\emptyset$ is given. The likelihood factor is 1, if the characteristics of this mating m_n location acquired in the subsequent iteration (*i+1*) through the utilization of local or global optimal mating attitudes are indistinguishable from the characteristics acquired in the previous iteration (*i*) using local or global optimal stances, then the likelihood factor is zero and the response remains the same.

As a crucial aspect of the collective, the variety of the group has a close association with the early convergence in the evolutionary process, the tardy pace in the subsequent evolution, and the inadequate convergence performance. Consequently, the diversity of the group plays an essential role in achieving global convergence of the algorithm. In light of this, the group's average position is determined based on the best position of its members. In statistics, the median and mean are commonly employed to indicate the general average level, with the mean being susceptible to the influence of extreme values.

$$P_m = \begin{cases} p_{\left(\frac{n}{2}\right)} + p_{\left(\frac{n}{2}+1\right)} & if\ n\ is\ even \\ p_{(n+1)/2}\ if\ n\ is\ odd \end{cases} \quad (10.1)$$

In which, *n* denotes the number of mayfly in groups. The group's average position is employed as the medium position in the male with velocity update algorithm and the expression from Eq. (2) is reformed as,

$$mv_{ij}^{t+1} = mv_{ij}^t + \sigma_1\, e^{-\gamma r_p^2}\left(sbest_{ij} - p_{ij}^t\right) + \sigma_2\, e^{-\gamma r_g^2}\left(cbest_j - p_{ij}^t\right) + \sigma_3\, e^{-\gamma r_g^2}\left(P_{mij} - p_{ij}^t\right) \quad (10.2)$$

However, during the beginning stage of spatial search in the swarm intelligence algorithm, the group with population is fairly fragmented in terms of distribution, and a few individuals have extremely poor positions. Therefore, using the mean since the average position for the group might not be an accurate representation of the average position for all groups. Hence, this overall mayfly work introduces the notion of the group with adapted mayfly (as in Eq. 10) based on the Gaussian concept, and medium position with their reformed position, which organises every mayfly individual according to the value of their goal function, then chooses the group's median position as the mayfly's median position.

Task scheduler

Tasks related to MRMPSO optimization must be formulated to improve the total and individual usefulness functions, and addressed using integrated optimization techniques. The computational results demonstrate that resource allocation, which is driven by task requirements, effectively reduces service delays and achieves remarkable energy efficiency. Therefore, it is a desirable solution for mobile cloud utilizations. The optimized scheduler and the entire process are illustrated in Figure 1 and Algorithm 1, respectively.

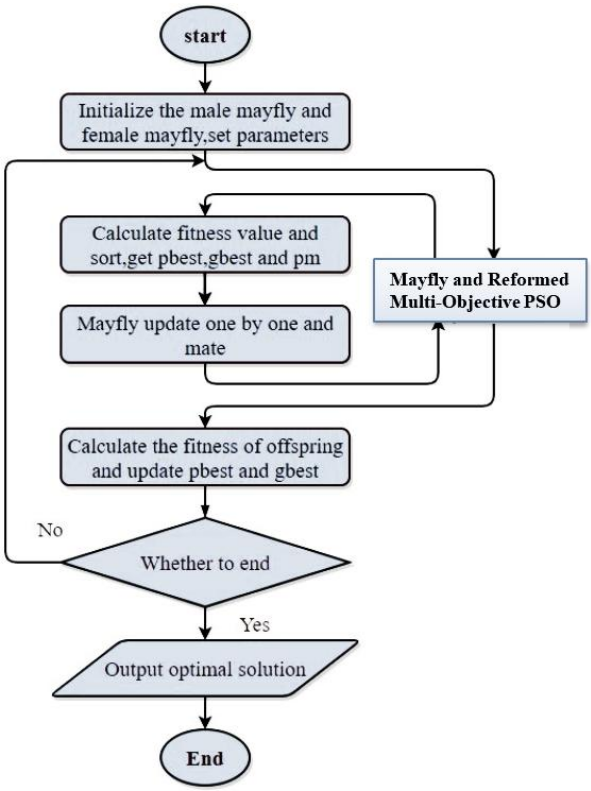


Fig 2. Mayfly and reformed multi-objective PSO flow diagram

One can suppose that a data-intensive MCC application T is represented as a collection of discrete activities. A model of an application is used as:

$$T = \{ts_1; ts_2; \ldots; ts_n\} \quad (11)$$

where n is number of tasks. Each task ts_i is modelled as:

$$ts_i = \{L_i; e_i; S_i; d_i\} \quad (12)$$

L_i denotes task input file location, remotely or locally; e_i as the quantity of task execution instructions; S_i as task input size; d_i task deadline. The location parameters and data size in various modeling tasks (split as low, medium, large and extra-large) have been included to achieve the goal of developing a data-aware MRMPSO optimisation model for allocating tasks for mobile applications in a hybrid MCC environment.

Resource Modeling

The system makes use of three different types of computing resources: cloudlets, public clouds, and mobile devices. A model of mobile device M_d is as follows:

$$M_d = \{\gamma; \delta_{cost}; \delta_{down}; \delta_{up}; e_d; d_s; \omega_p\} \quad (13)$$

δ_{down}; δ_{up} are download and upload bandwidth available in network; the cost of data communication δ_{cost} over a mobile network is represented by the following units: cost, mobile device storage as d_s, available device energy as e_d (Joule), available device memory as γ, and computing power of the device ω_p.

The mobile device (M_d) can offload computationally intensive tasks to the Ccl, which can perform the tasks more efficiently due to its higher processing power and larger storage capacity with processing cost (pc_{cost}) for cloud-based devices. A public cloud or cloudlet C_{cl} virtual machine can be modeled as:

$$C_{cl} = \{pc_{cost}; \delta_{cost}; \delta_{down}; \omega_{cl}\} \quad (14)$$

A cloud machine with processing power is represented by ω_{cl}, and its cost is represented by pc_{cost}. C is the expected overall financial cost of the MCC. The reformed multi-level PSO is utilized to allocate the resource for every MCC users depending on various tasks (split as low, medium, large and extra-large) as explained in algorithm 1.

Algorithm 1: Resource sharing modules

In: Input Network attributes;

Out: Decision rate for Resource allocation;

i. **for** every time slots,
ii. then **do**
iii. To evaluate the previous arrival of resource requests;
iv. Estimate the resource allocation in maximum;
v. Describe the MCC user access order;
vi. Repeat step 8. Initiate t = t+1;
vii. **for** t = 1 to T **do**
viii. T = T.t+1

ix. Repeat the resource allocation in cloud data;
x. Resolve the resource shortage and maximization problem;
xi. Evaluate size of queue in cloud;
xii. **If** (size of queue > 0)
xiii. then allot the resource for all MCC users; // multi-level cloud users
xiv. **Else**
xv. Allocate the resource for primary users;
xvi. Compute the overall load of the network;
xvii. Update the resource sharing details;
xviii. **End**

Energy and time consuming model

The total energy consumption (Eg^{Tot}) as the system anticipates that the compute environment will execute simultaneous tasks, MRMPSO in MCC is estimated by calculating the energy consumed by processing mobile device (Eg^{P}), the waiting energy (Eg^{W}), in particular whenever the local executing time is faster than the executing time in remote area. Eg^{c} is the mobile energy usage for communication and data transport in the portable gadget for job (t_j).

$$Eg^{Tot} = Eg^{P} + Eg^{C} + Eg^{W} \quad (15)$$

$$Eg^{P} = \sum_{j=1}^{x} \epsilon_j^{P} \quad (15.1)$$

$$Eg^{W} = (Mt^{W}) * \epsilon^{W} \quad (15.2)$$

$$Mt^{W} = maxi(0, (\sum_{j=1}^{x} d_{tj}^{P} - maxi(\sum_{j=1}^{c} d_{tj}^{P}, \sum_{j=1}^{cd}))d_{tj}^{P}) \quad (15.3)$$

$$Eg^{C} = \sum_{j=1}^{x} d_{tj}^{C} * \epsilon^{C} \quad (15.4)$$

$$D_{tj} = d_{tj}^{P} + d_{tj}^{C} + d_{tj}^{W} \quad (16)$$

$$d_{tj}^{P} = e_{j,targ} + (S_i * e_j) \quad (16.1)$$

$$d_{tj}^{C} = \frac{S_i}{mini(\beta_l, \beta_{ct}, \text{MiniCost})} + l \;\&\; d_{tj}^{W} = \frac{T_{rq}}{\mu} \quad (16.2)$$

In which, the specified parameters such as cd, c, x: describe the quantity of tasks to be performed in the cloudlet, public cloud, and the portable device, in corresponding order. $\sum_{j=1}^{x} d_{tj}^{C}$, $\sum_{j=1}^{x} d_{tj}^{P}$ & $\sum_{j=1}^{x} d_{tj}^{W}$ signify the overall processing duration for every tasks executed locally (mobile device) and remotely cloudlet and public cloud respectively. We take into account the energy consumption during waiting only if the waiting period Mt^{W} is positive value. ϵ_j^{P} is the predicted energy consumption of the mobile device for each of the following: task t_j, ϵ^{W} waiting for remote execution, and ϵ^{C} as data transfer/communication and the sensitivity factor for task-related data size is e_j. The mean typical rate at which a remote server receives requests to perform tasks is μ. The average amount of task requests currently in the queue is T_{rq}. Among the data location and the target estimation in the network bandwidth in minimum range is $mini(\beta_l, \beta_{ct})$ and their latency is l.

The algorithm presented below aims to provide the MRMPSO-based task scheduler with the most optimal resource allocation module. The fitness function is used to evaluate the optimisation value for a certain clarification depend on the total monetary/financial cost C (as per equation 14) in order to fulfil the goal of scheduling tasks on the specified compute environment and the total energy E (as per equation 15) consumed by the mobile device. The fitness function takes a reformed multi-objective PSO particle as input, where the position denotes a solution for scheduling application tasks. The estimation of energy and cost depend on the task execution time, including time of communication, processing, and waiting. The time taken for data communication (DC) for a task (tj) is influenced by the location of the task data. DC is only taken into account when the environments for data storage and computing are dispersed. DC time's effect on mobile device energy use and cost is assessed by the MRMPSO fitness function.

Algorithm 2: Task scheduling using MRMPSO based resource sharing

Input: tasks T, computation resources R

Output: task scheduler S

Upgrade resources with metadata
Set P (MRMPSO)
Initialize MRMPSO Particles P[NP] NP : number of mayflies (particles)
P:sbest = Init global best (sbest)
for i = 1 to NP - 1 do
Rand P[i]:POS
Fcost = PSOFFn(P[i]:POS; T)
UpdBestPos (P; P[i]; Fcost)
end for
Run PSO Iter
for i = 1 to NL - 1 do
for j = 1 to NP - 1 do

Estimate P[ij]:VY
Upgrade P[ij]:POS
FitCost = MRMPSOFFn(P[ij]:POS; T) //updated module with mayfly
UpdBestPos(P; P[ij]; Fcost)
end for
end for
MiniCost = PSOFFn(P:sbest:POS; T)
S = (P:sbest:POS;MiniCost)
return S

Algorithm 2 outlines the key steps involved in determining the best plan for scheduling tasks in resource-constrained mobile applications that are data-aware. The computation requirement and size of task ti are used to calculate the task processing time. The influence of data size is assessed by conducting experiments on tasks handling with varying data sizes. DP is then utilized to estimate the amount of energy required to carry out an activity locally and the expense of doing so remotely. When the mobile device is not in use, or when the total local execution is less than the maximum distant execution in a cloudlet or public cloud, the system nevertheless assumes additional energy consumption. The proposed MRMPSO based technique has been evaluated for performance using software simulation.

PERFORMANCE EVALUATIONS AND RESULTS

The proposed hybrid MRMPSO MATLAB is used to simulate the algorithm. Every task has its own login and run times, which were provided by Job shop software and underwent arbitrary initialization. Execution time, resource utilisation, reliability, make span, and throughput are other metrics examined in the findings investigation. Tables 1 and 2 define the parameters of the MCC in virtual environment [39] and the MRMPSO algorithm respectively.

Table 1 Parameters in Cloud system

Parameters	Range
Number of VMs	40–200
Amount of tasks	200–1000
Size of tasks (MI)	30000–200000
Power consumption (Watts)	200–1000
Execution rate (MIPS)	1000–5000
Percentage of power consumed in states from idle to active	0.6–0.7

Furthermore, the workload magnitudes were generated haphazardly at execution time and the extent is determined by Millions of Instructions (MI). Moreover, the investigation employed 80 servers with diverse resource capabilities and workloads. The size of the 800-1000 jobs that make up the extra-large assignment is between 100000 and 200000 MI. Similar to this, the major work is made up of 600–700 smaller assignments, each of which is between 70000 and 100000 MI in size. Similar to the little task, the medium-sized task is made up of 400–500 tasks with task sizes between 50000–70000 MI. The small-sized task is made up of 100–200 tasks that are between 30000 and 50000 MI in size [38-40]. Various scheduling methods have been chosen for comparisons, namely IPSO, Firefly and IPSO with the hybrid MRMPSO scheme.

Table 2 PSO considerations

Parameter	Value
Population size	70
Number of Iterations	100
Velocity boundaries	[−1, 1]

A comprehensive examination carried out between FIMPSO and prevailing techniques regarding memory usage is illustrated in Figure 3. This figure indicates that FIMPSO attained the highest memory utilization for all sorts of tasks when compared to alternative approaches. However, the model that was presented demonstrated better results as it achieved the highest level of memory usage for small tasks, which is 60%. However, when it comes to exceptionally large tasks, both prior models exhibited inadequate memory usage, with only minimal memory utilization being achieved. Thus, it can be deduced that the MRMPSO algorithm presented is extremely efficient in regards to memory usage across all task types, regardless of their magnitude. The diagram further indicates that an escalation in the quantity of tasks leads to a rise in memory consumption.

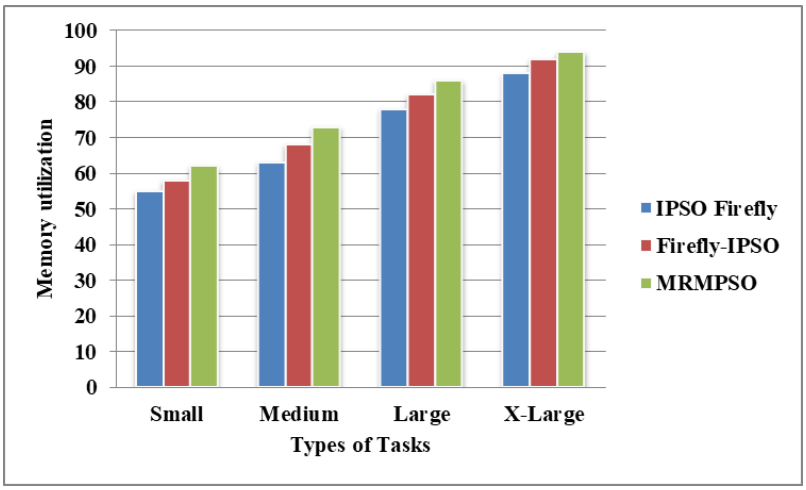


Fig. 3 Comparisons of memory usage for various tasks & algorithms

A comprehensive evaluation of the average throughput of different techniques is conducted, comparing MRMPSO with the currently available methods, as illustrated in Figure 4. Simultaneously, IPSO and FF-IPSO techniques endeavored to effectively handle the situation by achieving a marginal throughput in the mean data transfer rate of about 85% and 90%. However, the model that was presented demonstrated exceptional results by achieving the highest average throughput of 100% when handling extra-large tasks.

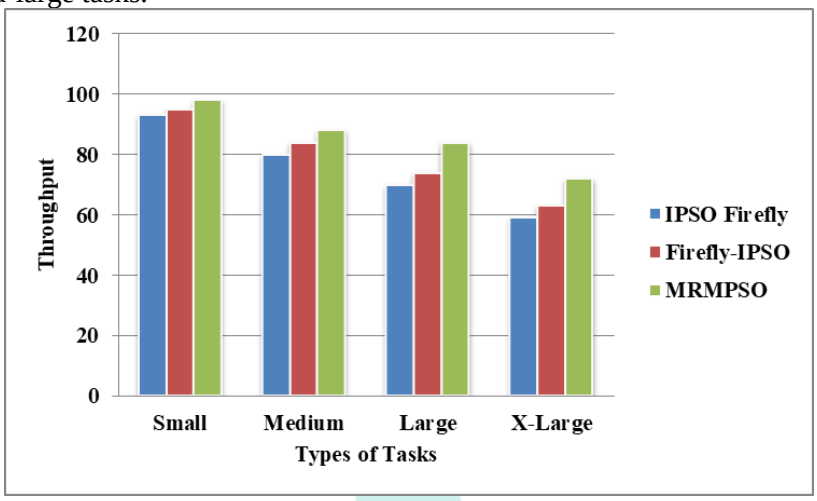


Fig. 4 Comparisons of average throughput for various tasks & algorithms

The dependability assessment of different techniques was conducted comparing MRMPSO and previously established techniques as depicted in Figure 5. In the case of minor tasks, conventional models exhibited inadequate dependability by attaining only the minimum level of reliability.

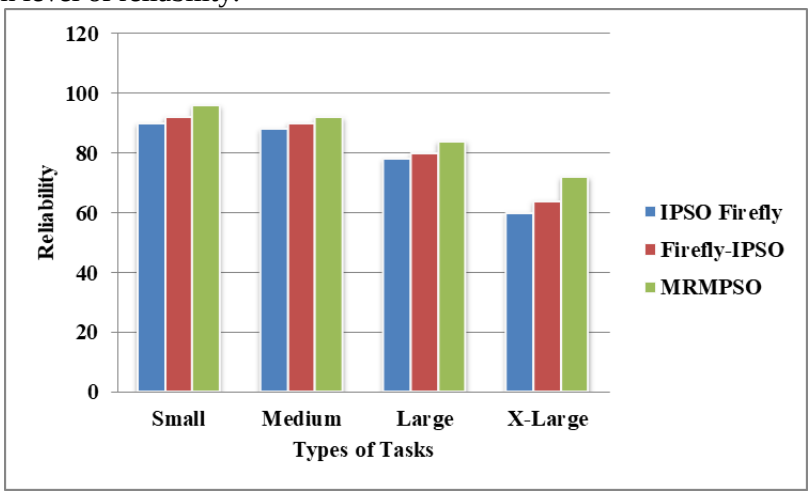


Fig. 5 Comparisons of reliability for various tasks & algorithms

Hence, it can be deduced that the MRMPSO algorithm introduced is extremely efficient in ensuring dependability across various task types, irrespective of their magnitudes. Additionally, it is evident that the dependability diminishes as the quantity of tasks rises.

Figure 6 presents an in-depth contrast of various scheduling techniques with the MRMPSO algorithm concerning the make span (completion time). The illustration demonstrates that the MRMPSO produced significantly superior outcomes compared to the contrasted scheduling techniques. Nonetheless, the MRMPSO algorithm put forward proved to be efficient, meeting the minimum make span requirement of about 145. It should be emphasized that the make span tends to augment as the number of tasks rises.

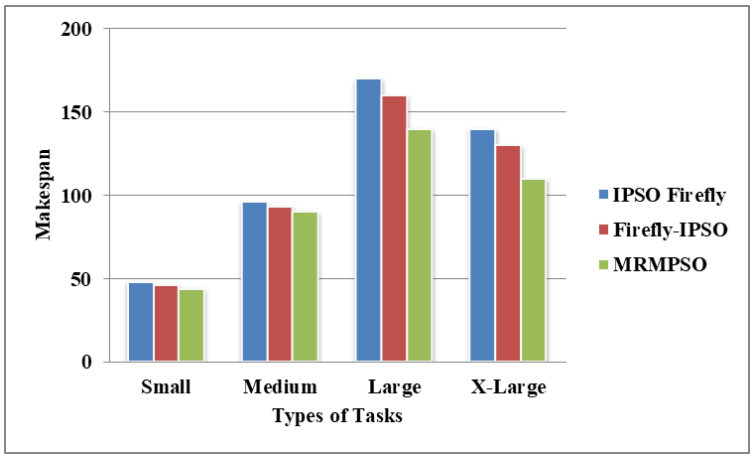


Fig. 6 Comparisons of make span for various tasks & algorithms

Table 3 presents an elaborate contrast of various scheduling techniques with the FIMPSO algorithm concerning the mean workload, mean turnaround time, and mean response time. The data in the table reveals that MRMPSO achieved significantly better outcomes than other scheduling algorithms.

Nonetheless, the MRMPSO algorithm put forth produced a favorable result with the minimal average processing time of about 18 ms. While gauging the outcomes in relation to the response time in average range, it is discernible that IPSO and FF approaches necessitated a maximum time of around 48 and 49 ms, respectively as depicted in Figure 7.

Table 3 Comparison of proposed MRMPSO and other scheduling methods

Methods	Average load	Average response time (ms)	Average turnaround time (ms)
IPSO [32]	0.45	48.23	56.994
Firefly [37]	0.47	49.18	56.44
FF-IPSO [40]	0.26	14.91	24.13
FIMPSO [38]	0.25	14.08	21.45
Proposed MRMPSO	0.24	13.18	18.88

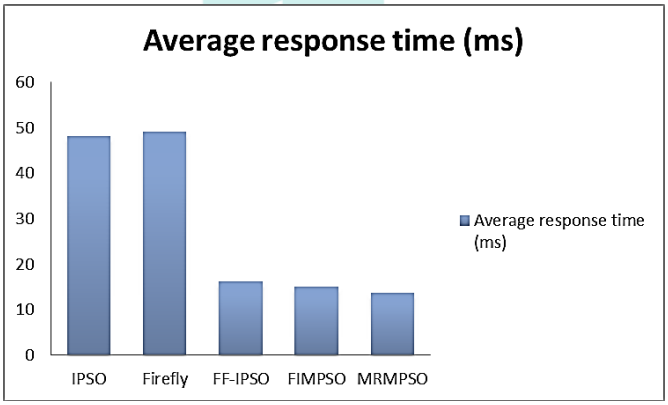


Fig. 7 Analysis of average response time analysis for various algorithms

Correspondingly, when evaluating the results based on the mean turnaround time, it is evident that IPSO and FF methods recorded the highest mean turnaround time of around 57 and 56 ms, respectively as depicted in Figure 8.

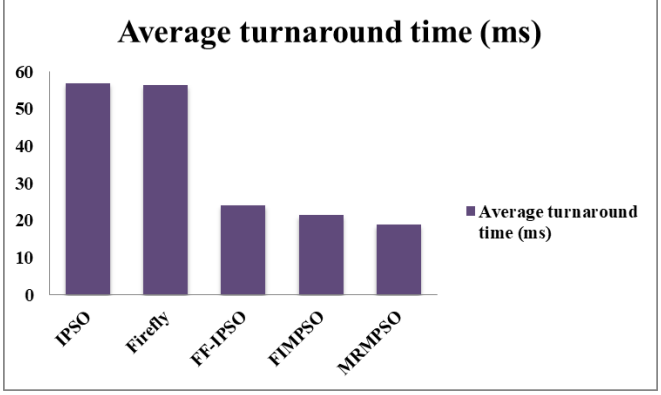


Fig. 8 Analysis of average turnaround time for various algorithms

Nonetheless, the suggested MRMPSO technique produced a successful result with the minimum mean response time of 13.18 milliseconds. Hence, it can be deduced that the MRMPSO algorithm put forward exhibits a commendable mean output rate across all task categories, regardless of their magnitudes. Additionally, it can be deduced that the mean output rate experiences a decline as the number of tasks rises. Also, it can be deduced that the MRMPSO algorithm demonstrated is exceedingly efficient when it comes to CPU usage for all task categories, regardless of their magnitudes. Additionally, the illustration suggests that augmenting the quantity of tasks further leads to a rise in memory utilization.

CONCLUSION

Based on the central concept of service-oriented utility functions, an architectural and mathematical framework for heterogeneous resource sharing is provided in this study. A unified paradigm is offered where all these qualities were equivalently mapped to time resources. This is because heterogeneous resources are frequently measured/quantified at dissimilar scales/units (for example, energy, latency, etc.) based on Mayfly and Reformed Multi-Objective Particle Swarm Optimization (RMPSO), referred to as MRMPSO. The origin of this mayfly phenomenon lies in the customary actions of mayflies, specifically their reproductive rituals. Our understanding is that mayflies reach maturity upon hatching and only the most resilient individuals endure, irrespective of their lifespan. The potential resolution to the challenging circumstances can be found in the positional coordinates of each and every mayfly in search spaces. The optimization problems are formulated with enhanced diverse parameters by utilizing service-oriented utility functions and resolved them by employing integrated optimization methods. Our numerical findings demonstrate that the utilization of service-oriented heterogeneous resource sharing effectively minimizes service latencies and attains exceptional energy efficiency, rendering it a desirable choice for mobile cloud deployment. The outcomes will be displayed in a real execution scenario to assess the variation in available bandwidth and network latency during practical execution scenarios such as 4G, 5Gs using the proposed module in the future.

REFERENCES

1. "Mobile Cloud Computing Forum," http://www.mobilecloudcomputingforum.com/.
2. Guan, L., Ke, X., Song, M., and Song, J., 2011, A survey of research on mobile cloud computing, Proc. of the 10th IEEE/ACIS International Conference on Computer and Information Science, pp. 387-392.
3. Dinh, H.T., Lee, C., Niyato, D., and Wang, P., 2011, A survey of mobile cloud computing: architecture, applications, and approaches, Wireless Communications and Mobile Computing.
4. Bifulco, R., Brunner, M., Canonico, R., Hasselmeyer, P., and Mir, F., 2012, Scalability of a mobile cloud management system, Proc. of the first edition of the MCC workshop on Mobile cloud computing, p. 17.
5. Bonomi, F., Milito, R., Zhu, J., and Addepalli, S., 2012, Fog computing and its role in the Internet of things, Proc. of the first edition of the MCC workshop on Mobile cloud computing, pp. 13-16.
6. W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy optimal mobile cloud computing under stochastic wireless channel," IEEE Trans. Wireless Commun., vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
7. C. You, K. Huang, H. Chae, and B. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," IEEE Trans. Wireless Commun., vol. 16, no. 3, pp. 1397–1411, Mar. 2017.
8. C. You, Y. Zeng, R. Zhang, and K. Huang, "Asynchronous mobile-edge computation offloading: Energy-efficient resource management," IEEE Trans. Wireless Commun., vol. 17, no. 11, pp. 7590–7605, Nov. 2018.
9. S. Huang, B. Lv, R. Wang, and K. Huang, "Scheduling for mobile edge computing with random user arrivals: An approximate MDP and reinforcement learning approach," IEEE Trans. Veh. Technol., vol. 69, no. 7, pp. 7735–7750, Jul. 2020.
10. C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," IEEE Trans. Commun., vol. 67, no. 6, pp. 4132–4150, Jun. 2019.
11. S.-W. Ko, K. Huang, S.-L. Kim, and H. Chae, "Live prefetching for mobile computation offloading," IEEE Trans. Wireless Commun., vol. 16, no. 5, pp. 3057–3071, May 2017.
12. W. Chang, Y. Xiao, W. Lou, and C. Shou, "Offloading decision in edge computing for continuous applications under uncertainty," IEEE Trans. Wireless Commun., vol. 19, no. 9, pp. 6196–6209, Sep. 2020.
13. P. Jain, J. Manweiler, and R. R. Choudhury, "Low bandwidth offload for mobile AR," in Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT), 2016, pp. 237–251.
14. G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," IEEE Commun. Mag., vol. 58, no. 1, pp. 19–25, Jan. 2020.
15. Abolfazli, S., Sanaei, Z., Gani, A., Xia, F., Yang, L.T.: Rich mobile applications: genesis, taxonomy, and open issues. Journal of Network and Computer Applications 40, 345-362 (2014).
16. Chun, B.G., Ihm, S., Maniatis, P., Naik, M., Patti, A.: Clonecloud: elastic execution between mobile device and cloud. In: Proceedings of the sixth conference on Computer systems. pp. 301{314. ACM (2011).
17. Cuervo, E., Balasubramanian, A., Cho, D.k., Wolman, A., Saroiu, S., Chandra, R., Bahl, P.: Maui: making smartphones last longer with code ooad. In: Proceedings of the 8th international conference on Mobile systems, applications, and services. pp. 49-62. ACM (2010).
18. Kemp, R., Palmer, N., Kielmann, T., Bal, H.: Cuckoo: a computation offloading framework for smartphones. In: International Conference on Mobile Computing, Applications, and Services. pp. 59-79. Springer (2010).
19. Giurgiu, I., Riva, O., Juric, D., Krivulev, I., Alonso, G: Calling the cloud: enabling mobile phones as interfaces to cloud applications. In: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware. p. 5. Springer-Verlag New York, Inc. (2009).

20. March, V., Gu, Y., Leonardi, E., Goh, G., Kirchberg, M., Lee, B.S.: **μ** cloud: towards a new paradigm of rich mobile applications. Procedia Computer Science 5, 618-624 (2011).
21. Kosta, S., Aucinas, A., Hui, P., Mortier, R., Zhang, X.: Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: Infocom, 2012 Proceedings IEEE. pp. 945-953. IEEE (2012).
22. Rahimi, M.R., Venkatasubramanian, N., Mehrotra, S., Vasilakos, A.V.: Mapcloud: mobile applications on an elastic and scalable 2-tier cloud architecture. In: Proceedings of the 2012 IEEE/ACMfth international conference on utility and cloud computing. pp. 83-90. IEEE Computer Society (2012).
23. Sanaei, Z., Abolfazli, S., Gani, A., Shiraz, M.: Sami: Service-based arbitrated multi-tier infrastructure for mobile cloud computing. In: Communications in China Workshops (ICCC), 2012 1st IEEE International Conference on. pp. 14-19. IEEE (2012).
24. Nan, X., He, Y., Guan, L.: Optimal resource allocation for multimedia cloud based on queuing model. In: Multimedia signal processing (MMSP), 2011 IEEE 13th international workshop on. pp. 1{6. IEEE (2011).
25. Zhou, B., Dastjerdi, A.V., Calheiros, R., Srirama, S., Buyya, R.: mcloud: A context-aware offloading framework for heterogeneous mobile cloud. IEEE Transactions on Services Computing (2015).
26. Armbrust, M., Fox, A., Grith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Communications of the ACM 53(4), 50-58 (2010).
27. Satyanarayanan, M., Bahl, V., Caceres, R., Davies, N.: The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing (2009).
28. D. Chaudhary, B. Kumar, An analysis of the load scheduling algorithms in the cloud computing environment: A survey, in: IEEE 9th International Conference on Industrial and Information Systems, ICIIS, IEEE, 2014, pp. 1–6.
29. J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, vol. 4, IEEE, 1995, pp. 1942–1948.
30. C. Gupta, S. Jain, Multilevel fuzzy partition segmentation of satellite images using GSA, in: International Conference on Signal Propagation and Computer Technology, ICSPCT, 2014.
31. E. Pacini, C. Mateos, C.G. Garino, Distributed job scheduling based on swarm intelligence: A survey, Comput. Electr. Eng. 40 (2013) 252–269.
32. A.E.M. Zavala, A.H. Aguirre, E.R.V. Diharce, S.B. Rionda, Constrained optimization with an improved particle swarm optimisation algorithm, Int. J. Intell. Comput. Cybern. 1 (3) (2008) 425–453.
33. S. Pandey, R. Buyya, et al., A particle swarm optimization based heuristic for scheduling workflow applications in cloud computing environments, in: 24th IEEE International Conference on Advanced Information Networking and Applications, IEEE, 2010, pp. 400–407.
34. Mo, S.; Ye, Q.; Jiang, K.; Mo, X.; Shen, G. An improved MPPT method for photovoltaic systems based on mayfly optimization algorithm. Energy Rep. 2022, 8, 141–150.
35. Z. Liu, P. Jiang, J. Wang, and L. Zhang, ``Ensemble forecasting system for short-term wind speed forecasting based on optimal sub-model selection and multi-objective version of mayfly optimization algorithm," Expert Syst. Appl., vol. 177, Sep. 2021, Art. no. 114974.
36. J. Gupta, P. Nijhawan, and S. Ganguli, ``Parameter estimation of fuel cell using chaotic mayflies optimization algorithm," Adv. Theory Simul., vol. 4, no. 12, Dec. 2021, Art. no. 2100183.
37. Alsaidy, S.A., Abbood, A.D. and Sahib, M.A., 2022. Heuristic initialization of PSO task scheduling algorithm in cloud computing. *Journal of King Saud University-Computer and Information Sciences*, *34*(6), pp.2370-2382.
38. Devaraj, A.F.S., Elhoseny, M., Dhanasekaran, S., Lydia, E.L. and Shankar, K., 2020. Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. *Journal of Parallel and Distributed Computing*, *142*, pp.36-45.
39. Z. Fan, T. Wang, Z. Cheng, G. Li, F. Gu, An improved multi-objective particle swarm optimization algorithm using minimum distance of point to line, Shock Vib. (2017) (2017).
40. Golchi, M.M., Saraeian, S. and Heydari, M., 2019. A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation. *Computer Networks*, *162*, p.106860.