International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Dynamic Multi-Objective Task Scheduling Scheme in Mobile Cloud Computing

Rameshwar Singh Sikarwar[1], Dr. Rajeev G. Vishwakarma[2]

Submitted: 17/05/2024 **Revised:** 25/06/2024 **Accepted:** 04/07/2024

Abstract: By shifting computation-intensive activities to distant cloud servers, The concept of Mobile Cloud Computing (MCC) has emerged as a prospective paradigm that has the ability to enhance the computing capabilities of mobile devices that are limited in their resources. The performance of MCC systems may be significantly improved by the use of task scheduling, which helps to maximize the utilization of resources while simultaneously minimizing energy consumption and latency. We provide a unique multi-objective task scheduling approach designed specifically for MCC contexts in this research. The suggested plan seeks to balance a number of competing goals, such as resource usage, energy consumption, and job completion time. To efficiently assign tasks to suitable cloud and mobile resources, We provide a solution that is based on heuristics and represent the task scheduling issue as a multi-objective optimization problem. The usefulness and superiority of the suggested system over current methods in terms of achieving better trade-offs among competing objectives are demonstrated by experimental assessments.

Keywords: *Mobile Cloud Computing, Task Scheduling, Multi-Objective Optimization, Energy Consumption, Resource Utilization, Heuristic Algorithms.*

1. Introduction

With their many features and services, mobile devices like smartphones and tablets have evolved into essential tools in today's world. Yet, these devices' incomplete computing properties (such as processing speed, recollection, and battery life) frequently make it difficult for them to quickly complete complicated tasks and services. In order to get around these restrictions, a potential paradigm known as Mobile Cloud Computing (MCC) has surfaced. MCC uses the processing power of distant cloud servers to offload computation-intensive activities from mobile devices. Task scheduling, which efficiently distributes work across available cloud and mobile resources, is essential for maximizing the performance of MCC systems. However, because of the variability of resources, fluctuating network conditions, and dynamic nature of mobile apps, task scheduling in MCC contexts is difficult. Furthermore, conventional task scheduling techniques frequently concentrate on maximizing a particular goal (such as reducing energy usage or job completion time), which may result in less-than-ideal solutions that overlook other crucial performance indicators. A number of virtualized resources are achieved through cloud computing, making scheduling an important factor. For a client to complete the operation in the cloud, many virtualized resources could be required. For this reason, manual scheduling is not possible. It is necessary to automatically simulate the scheduling process using scheduling algorithms that seek to minimize the virtual machine's energy consumption and increase CPU usage. Within the sphere of cyber-physical systems, MCC is now a field that shows great promise. The cloud computing and mobile computing have been combined into this one.

The computational offloading feature of the mobile cloud application is one of its distinguishing features. This feature allows the work to be sent to the cloud server, where it is processed, and then the results are returned to the mobile device. During the offloading process, the job has to be queued up on cloud servers and divided up across virtual machines. The process of

1,2Department of Computer Science and Engineering
1Research Scholar, Dr. A. P. J. Abdul Kalam University, Indore
2Research Supervisor and Pro Vice-Chancellor, Dr. A. P. J. Abdul Kalam University, Indore, (M.P.), India
Corresponding Author : Rameshwar Singh Sikarwar
Email : rameshwarsingh@aku.ac.in

assigning and processing mobile tasks to servers involves a crucial phase called task scheduling. An important consideration in the unloading process as a whole is energy conservation. For the purpose of scheduling, the task consists of mapping the work that has been offloaded to the cloud server while adhering to time and energy limits. The purpose of this chapter is to provide a hybrid scheduling strategy that is based on bacterial foraging optimization (BFO) and Gaussian-based multi-objective particle swarm optimization (GMOPSO). This particular PSO system has superior performance in terms of both its makespan and its energy efficiency as compared to earlier iterations of the system.. Within the cloud service providers, the cloud schedulers are a completely managed organization. It offers a dependable solution and reduces the need for human participation in work scheduling. The jobs are planned over a range of virtual machines housed within the data centers' real servers.

One of the most important aspects of cloud computing is virtualization [1], which enables many operating systems to be run concurrently on a single physical machine (PM). Simulations of machinery, a specific type of middleware, allow various operating systems to operate independently of one another on these physical machines. Hypervisors are the middleware programs that allow these virtual machines to be created, run, and managed on one or more pools of real computers. Scheduling on the virtual machines of the real system is made possible by the brokers or hypervisors. The cloud service providers maintain track of all the requests made to the server and the amount of physical machine utilization that occurs within the data center. The task's attributes are noted, and dependencies are strictly monitored as well. The two main categories of tasks that are sent to the server are input-output (IO) bounded and CPU bounded. To process the compute-intensive operations, a lot of RAM is needed. The peripheral devices linked to the servers are mostly needed for IO-based functions.

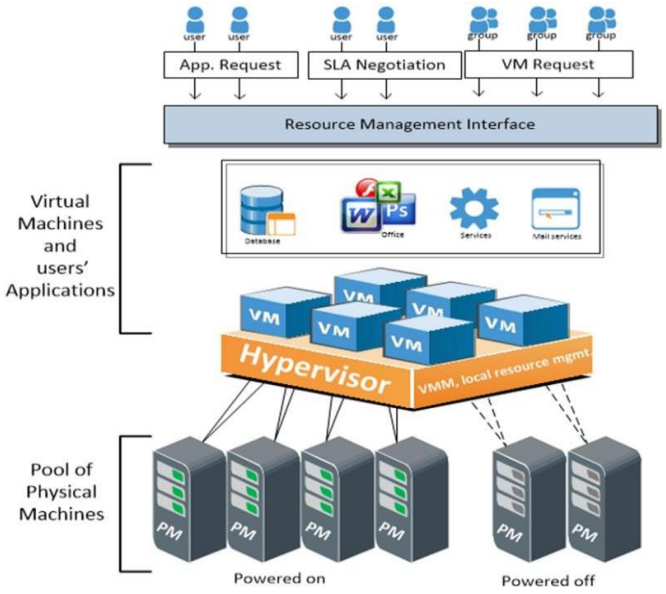

Figure 6-1 Virtualization in cloud computing

An innovative multi-objective task scheduling system that is specifically designed for MCC contexts is presented by us in this study. An effort has been made to maximize a number of competing goals, such as the amount of time it takes to do a work, the amount of energy that is used, and the amount of resources that are utilized. The task scheduling issue is formulated as a multi-objective optimization problem, and a heuristic-based technique is developed in order to effectively distribute jobs to the appropriate cloud and mobile resources. In terms of establishing better trade-offs among competing goals, experimental assessments and results reveal that the suggested method is both successful and superior to other ways that are already in use.

2. Related Work

The literature has reviewed task scheduling in MCC

contexts in great detail, and many strategies have been put forth to handle the particular difficulties this field presents. For work scheduling in MCC systems, early research efforts concentrated on heuristic-based algorithms like Round Robin and First Come First Serve (FCFS). Nevertheless, these methods frequently perform less well than ideal when it comes to energy and resource usage.

More advanced task scheduling methods for MCC settings are the product of current advancements in optimization strategies, which led to their development. To handle the trade-offs between competing objectives in task scheduling, multi-objective Numerous optimization strategies, such as ant colony optimization, particle swarm optimization, and genetic algorithms, have been used on a regular basis. These methods look for Pareto-optimal answers that strike a balance between several performance indicators. Moreover, task scheduling in MCC systems has been enhanced by the use of machine learning methods such as deep learning and reinforcement learning, which are used to develop the most effective scheduling algorithms based on past data and system dynamics. In dynamic MCC contexts, these strategies have demonstrated encouraging outcomes in terms of enhancing the scalability and flexibility of task scheduling algorithms. Even with these developments, additional research is needed to create job scheduling schemes that are more effective and efficient, able to manage the complexity of MCC systems and maximize several objectives at once.

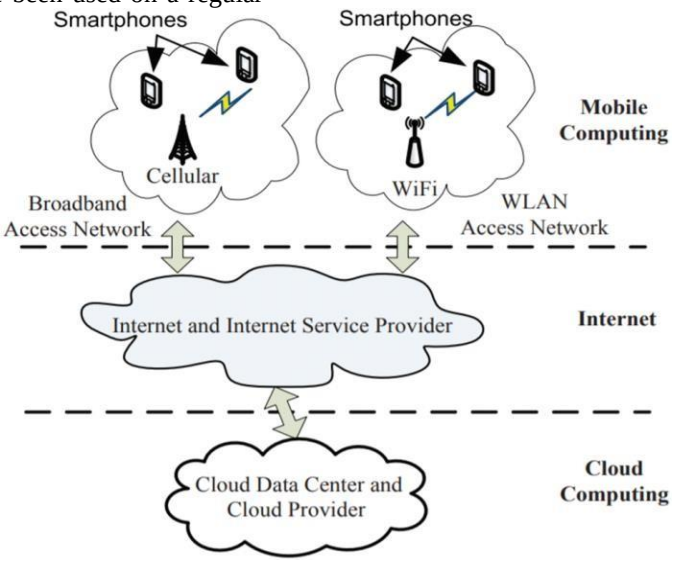

Figure 2 System model for scheduling the offloaded task to the cloud server

3. Task scheduling during computational offloading

Within mobile cloud computing, computation offloading is a mechanism where an application is divided into local and remote execution based on certain parameters. The system model for offloaded task task scheduling is shown in Fig.1. An application is divided throughout the offloading process, and depending on a number of factors, the choice to do the job locally or offload it has been made. It is used to offload tasks that are determined to be completed on an edge server. Minimizing the mobile task's makespan and energy cost are the two objective functions taken into consideration in the job.

Once these jobs arrive to the cloud server, a scheduling approach is used to schedule them. When it comes to mobile cloud computing, one of the most significant roles is the scheduling of tasks on the cloud server. When it comes to the execution of the job, the cloud service provider is responsible for assigning the virtual machines (VM). Research in mobile computing has benefited greatly from frameworks such as Chroma(RPC) [2], Cuckoo(RMI) [3], spectra[4], MAUI [5], Mobicloud [6], and Clonecloud [2,7]. These are well-liked infrastructures within the realm of cloud computing that provide reinforcement for the concept of job offloading to the cloud server, either through task division or by taking an entire application into consideration.

Previous research has been carried out in a number of different studies in an effort to optimize various target functions, including makespan, energy, load balancing, the cost, as well as the quality of service (QoS). Considering that task scheduling is an NP-hard issue, there is a lot of room for optimization. A cloud is described as the interdependence among the several computing jobs that make up the mobile application, which are represented as nodes. These delegated computational activities demand resources on the cloud servers to be completed. The cloud service providers must guarantee the availability of these resources, and service prices may differ across national borders.

There are two ways to carry out the work under the mobile task offloading model: either locally on the mobile phone or outsourcing the work to a server located in the cloud. Immediately after the first step of task splitting, the decision engine gathers numerous device and network information through the profiling process before deciding whether to offload. Now the job gets to the cloud server over a Wi-Fi or cellular network. Offloading is the process of moving The calculation is sent to a server that is both resourceful and situated at a distance in order to improve the performance of the device and decrease the amount of energy that is used. In order to determine whether or not to offload a decision to a remote server, it is necessary to take into consideration the many elements that influence the performance of the device. It is also possible to do partial offloading in certain circumstances. A mobile device is responsible for handling a part of the application work, while the remaining piece is being sent to a cloud or surrogate server location. The length of time it takes to calculate the work is determined by the amount of computing that is required as well as the processing speed of the mobile device used. Take into consideration, for the sake of this scenario, that the job is divided into two parts, the first of which is executed locally, and the second of which is executed on a server located at a distance. Consider the following: let CA_LOCAL represent the quantity of the calculation, let PS_LOCAL represent the processing speed of the mobile device, and let CT_LOCAL represent the amount of computation time required on the local device for local execution. These principles will be related to one another in the following manner:

$$C_{T\_LOCAL} = C_{A\_LOCAL} / P_{S\_LOCAL} \quad (1)$$

The cloud or edge server is used to execute the second partition, which is used for remote execution. The amount of data that is going to be transferred is denoted by DAMOUNT_OF_DATA, while the bandwidth that is available on the device is denoted by BAVAILABLE. CT_REMOTE is the maximum amount of time that must pass before the data may be sent to and from the server.

$$C_{T\_REMOTE} = D_{AMOUNT\_OF\_DATA} / B_{AVAILABLE} + Cloud\ processing\ time \quad (2)$$

In order to calculate the total amount of time that is necessary to execute the program both locally and remotely, the sum of the two equations that were presented before will be used. This is

$$C_{T\_TOTAL} = C_{T\_LOCAL} + C_{T\_REMOTE} \quad (3)$$

When a task is sent from a mobile device to a cloud server, it is transmitted to the server that is owned and operated by the cloud service provider. It is the responsibility of the cloud service provider to retain a record of every single detail about the task that was handed over to them for completion. The cloudlets (task) are assisted in assigning the virtual machines by the Datacenter Broker policy [8]. The policy of the data center has to be appropriate for the maximum amount of time that the cloudlet may be executed. A mobile application, just like an internet application, is composed of a number of different activities. These activities may be represented as a directed acyclic graph (DAG), which is one method to see them. There are a number of virtual machines that are capable of running the program's separate job simultaneously; however, the dependent jobs must be synchronized according to their order of priority.

4. Model of the System and Formulation of the Problem

This section provides an overview of the system concept as well as the difficulty of work scheduling in settings involving MCC organizations. We examine a situation in which a collection of jobs with different computing demands are delivered to a mobile device for completion. The mobile device may transfer work to distant cloud servers for processing, but its battery life and computational power are constrained. The objective is to have less tasks. completion time, energy

usage, and resource contention by scheduling jobs to available cloud and mobile resources.

Consider the collection of tasks that need to be planned, denoted by the symbol \(T = \{t_1, t_2,..., t_n\}\), where each task \(t_i\) has a deadline, denoted by \(d_i\) and a computational requirement, \(c_i\). The mobile device has \(m\) available processing cores and a limited battery capacity, denoted by \(B\). Additionally, there are \(n\) cloud servers available for task offloading, each with a processing capacity, \(C_j\), and energy consumption rate, \(E_j\).

The collection of tasks that need to be planned is denoted by the symbol \(T = \{t_1, t_2,..., t_n\}\) where each job \(t_i\) has a deadline represented by the symbol \(d).The issue of work scheduling may be expressed as a multi-objective optimization problem with the following objectives, if one so chooses.

1. Decrease the amount of time required to finish a task (also known as "makespan").

2. Reduce the amount of energy you utilize.

3. Make the most efficient use of the resources that are available

The following mathematical formulation may be used to the issue of scheduling tasks with multiple objectives.

[The expression f_1(\textbf{x}) = \text{Makespan}(\textbf{x})] is a minimalization operation.

f_2(~textbf{x}) = \text{Energy Consumption}(\textbf{x})] is the formula for minimizing energy consumption.

f_3(x) = f_3(Resource use)(x) The expression "maximize" is used to maximize the use of resources.

Taking into consideration the following: \[\text{Makespan}(\textbf{x}) \leq d_i, \forall t_i \in T]

[The total of c_i is less than or equal to B] [The sum of x_i is less than or equal to C_j, for all t_i being inside T]

with \(\textbf{x}\) standing for the task allocation matrix, \(x_{ij}\) being a binary variable that indicates whether or whether task \(t_i\) is allocated to cloud server \(j\), and \(f_1\), \(f_2\), and \(f_3\) being the objective functions. _i is the variable that represents the task allocation matrix..

5. Proposed Multi-Objective Task Scheduling Scheme

This section provides an explanation of the multi-objective task scheduling method that is generally recommended for use in MCC situations. The plan might be broken down into the following phases.:

1. Task Prioritization: Arrange tasks according to their computational needs and deadlines.

2. Resource Allocation: Using a heuristic-based methodology that takes into account the trade-offs between competing goals, assign tasks to available cloud and mobile resources.

3. Dynamic Adaptation: Adjust the job scheduling plan dynamically in response to current system circumstances and resource availability.

Tasks with tighter deadlines and more computational demands are prioritized for scheduling, thanks to the task prioritization phase. This makes it easier to fulfill deadlines and reduce the total amount of time needed to complete the assignment. The multi-objective technique aims to optimize many objective functions simultaneously [9]. For the sake of the investigation, makespan and energy are regarded as objective functions. Reducing both functions is the real purpose of a task offloading method, which is based on maximizing the multi-objective function. The time needed for the task CPU to process and send the data is known as the makespan. When determining a task's makespan, the virtual machine's processing capacity and the amount of the job are both taken into account. This equation may be used to describe it.:

Makespan(T) = size of the task / computational power (4)

Energy costs are determined by two factors: the second is the calculation of virtual machine consumption charges, which vary depending on the cloud service provider. The task's execution time is the other. It may be described by the equation that follows:

Energy Cost = execution time X virtual machine usage charge (5)

The combination of bacteria foraging optimization (BFO) with particle swarm optimization (PSO) forms the basis of the suggested technique (GMOPSO-BFO).

While the BFO functions best with local search capabilities, the PSO technique performs very well while looking for the answer worldwide. When these two strategies are combined, the result is a more globally and locally optimum solution with a faster convergence time.

The size of the job and the characteristics of the virtual machine affect how long a given task takes to complete in this study. The following are some fundamental definitions related to task scheduling on mobile devices:

Assume the following:

a) A job of the application tasks T = {T1, T2…, Tx};

b) A collection of n virtual machines V = {V1, V2, V3…, Vn}

c) Ti and Tj are any two jobs, and E is the collection of links between them.

d) The data center's collection of physical machines (PMs) = (PM1, PM2, PM3..., PMn)

The research assumes that there is sufficient processing power available to the cloud service provider. Processing units (CPU), random access memory (RAM), and networking capabilities vary throughout virtual machines. V virtual machines are installed on the actual computers. Once the job is near, the data center brokers allocate the machine to it after keeping an eye on all the resources that are accessible. Every work that needs processing resources must wait in line, and tasks are scheduled to run on the system according to a task scheduling scheme[10].

5.1 Bacteria Foraging Optimization

In the human body's E. Coli (intestine), bacteria have a natural tendency to seek out and gather food in order to survive. This process is known as the "bacteria foraging method" [11]. Bacteria's primary survival strategy is to locate the nutrients, manage them, and consume the food in order to receive the energy needed to survive and proliferate. Typically, the system eliminates bacteria that cannot effectively scavenge for the nutrient. The concept of "survival of the fittest" is upheld." The scientists were intrigued by the idea of evolution and were inspired to apply it as an optimization technique. Such an evolutionary technique may be used for the majority of optimization procedures. The bacteria's primary goal is to obtain the most energy possible in a given amount of time while foraging. It is dependent upon a number of variables, such as the environment's features and the density of prey. It also relies on how perceptive and intelligent the bacteria are.

The cell structure of E. coli bacterium consists of a nucleoid, ribosomes, cytoplasm, pilus, and plasma membrane, among other biological components. While these characteristics carry out regular cellular functions, another essential characteristic—the flagellum—allows bacteria to proliferate or travel in various directions. The process by which an organism moves Chemotaxis is the process by which an object moves from its position in the presence of chemical attractants and repellents.. Two different movements are made possible by flagella: one is clockwise movement, which is known as rolling, and the other is counterclockwise movement, which is known as swimming. The tumbling and swimming movements of bacteria in E. Coli are shown in Fig. 3. It swims in an environment that is favorable for it, one in which there are enough nutrients available and the gut is neither acidic nor alkaline[12]. Usually, it tumbles, which causes the swim to change direction.

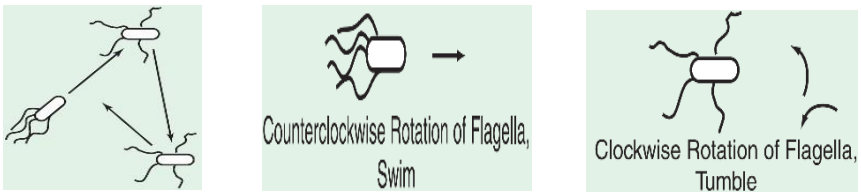

Figure. 3 Chemotaxis process of the bacteria

Swarming is another important bacterial mechanism in which bacteria produce attractants to group together in search of food. If the attractants are released deep and high, many bacteria may examine food together; in the

opposite scenario, they may examine food alone. In order to multiply, the bacteria divide into two during the reproductive process. Based on their fitness function or the nutrients they have access to, bacteria proliferate[13]. Because of their local surroundings, the bacteria also go through the elimination and dissemination phases of their lives. The ability to live can occasionally decline after a sharp increase in heat or nutrition. In terms of computation, the removal and dispersal procedure helps prevent being stuck in the local optima.

Optimization of particle swarms (PSO)

Nature serves as the inspiration for the particle swarm optimization (PSO) approach [14]. [12,15] It is based on the social behavior and dynamic movement of a flock of birds..A swarm is a collection of birds that travel together while looking for food at varying speeds and directions. Every bird or particle searches for food, and other birds often follow suit. During their quest, these birds converse with one another and usually follow one another closer to the meal[3,16]. After a set amount of time, the proximity from the meal is computed as a fitness value. Every bird within the flock is shown as a particle in three dimensions, possessing a specific speed and location. Each particle has two locations recorded in its memory: its individual best position (pbest) and the group's global best position (gbest). The formula is used to update the particle's velocity in the traditional PSO.

$$v_i^{(k+1)} = \left[\omega v_i^{(k)} + c_1\xi_1(b_i^{(k)} - x_i^{(k)}) + c_2\xi_2(y^{(k)} - x_i^{(k)})\right] \quad (6)$$

The constriction factor in the revised PSO version, which increased the convergence rate, was where the velocity vector as

$$v_i^{(k+1)} = \chi\left[v_i^{(k)} + c_1\xi_1(b_i^{(k)} - x_i^{(k)}) + c_2\xi_2(y^{(k)} - x_i^{(k)})\right] \quad (7)$$

The optimal place that the ith particle visited is indicated by b(k), where v(k-1) is the particle's velocity, x(k) is the ith particle's location at step k, and ↄ is a constriction factor in the equation above. Overall, y(k) is the best place I've ever been. It has been noted that after the integration of the Gaussian The density function of the aforementioned equation produces better results in terms of the global solution. The randn and Randn will be based on the absolute value of the Gaussian density function in the updated velocity equation. The Gaussian random density function is represented as

Pseudocode GMOPSO-BFO approach for task scheduling:

Initialize the Bacteria Foraging Optimization (BFO) parameters and Particle swarm optimization (GMOPSO) parameters:

N_p, N_c, S_l, N_r, N_e, C, $P_{dispersal}$, $d_{attract}$, $w_{attract}$, $h_{attract}$, $w_{attract}$, p_i, f, v_i

Input: a collection of all bacteria where each bacteria represented as $\theta^i(j,k,l)$

Output: a collection of information on how much these bacteria collect nutrients

1. **begin:** Let $\theta^i(j,k,l)$ be the position of the i^{th} bacteria in the environment where *j* defines the chemotaxes step, *k* defines the reproduction step, and *l* defines the dispersal elimination step.
2. for all bacteria in the list:
3. **Loop** elimination-dispersal step
4. **Loop** reproduction step
5. **Loop** Chemotaxis step
6. go for chemotactic steps using (a) and (b), respectively

7. Initialize the value of v_i and position p_i of the i^{th} bacteria

(a) Compute tumbling step:

$$\theta^i(j+1,k,l)=\theta^i(j,k,l)+C(i)\frac{dlt(i)}{\sqrt{dlt^T(i)dlt(i)}}$$

(b) Compute Swim step:

$$J(i,j,k,l)=J(i,j,k,l)+J_{cc}\left(\theta^i(j,k,l),P(j,k,l)\right)$$

8. Set J_{last} = $J(i,j,k,l)$

9. If $J(i,j+1,k,l)$ < J_{last}

10. Update J_{last}

11. For the reproduction phase: calculate the fitness function using:

$$J^i_{health}=\sum_{j=1}^{N_c+1}J(i,j,k,l)$$

12. Sort in ascending order the bacteria and chemotactic parameters

If $(k < N_r)$, perform reproduction step again till $k = N_r$

13. For elimination and dispersal:

14. for each bacteria,

15. if ($p_{ed} < P_{dispersal}$),

16. do elimination and dispersal till $l = N_e$

17. Do Mutation of the remaining bacteria (particles) using PSO scheme

18. Update p_i, best and g_i, best upon meeting the condition

$$p_{i,best} = p_i \quad \text{if } f(p_i) > f(p_{i,best})$$
$$g_{i,best} = g_i \quad \text{if } f(g_i) > f(g_{i,best})$$

19. Update velocity of each bacteria (particle) after every iteration by the Gaussian based velocity-

$$v_i^{(k+1)} = \left[\| randn | (b_i^{(k)} - x_i^{(k)}) + Randn | (y^{(k)} - x_i^{(k)}) \right]$$

20. Update position of each bacteria (particle) after every iteration by the formula-

$$x_i^{(k+1)} = x_i^{(k)} + v_i^{(k+1)}$$

21. Check p_i, which should exist within the range
22. Repeat step reproduction and PSO until convergence is achieved.
23. After the stopping criteria are met, the value of g_{best} and $f(g_{best})$ must be recorded.
24. End

---

6. Performance and Evaluation

6.1 Experimental setup

The suggested method was created using the programming language Python in the Windows 10 environment on an Intel Core i5 processor running at 1.80 GHz and 8 gigabytes of central processing unit memory [17,18]. In table 6-1, you will find a presentation of the many parameters that were taken into consideration during the simulation of the suggested method. For the purpose of analyzing the suggested technique, five virtual machines are taken into consideration, and a collection of jobs ranging from one hundred to one thousand is assumed. The outcomes are evaluated in relation to Using Python in the Windows 10 environment, an Intel Core i5 processor operating at 1.80 GHz and 8 gigabytes of random access memory (RAM) were used to develop the solution that was recommended. In order to simulate the recommended procedure, the many factors that were taken into consideration are listed in Table 6-1. During the process of evaluating the proposed method, five virtual machines are taken into consideration, and it is assumed that there are a total of one hundred to one thousand tasks. A comparison is made between the findings and the existing body of research on MOPSO [18] and BFO [19,20,21] with regard to the duration of time required to complete a job and the amount of energy that is saved. The BFO and Gaussian swarm techniques, both of which were used in MOPSO[21,22], serve as the foundation for the approach that has been proposed.For the sake of the experiment, the beginning size of the PSO is set at twenty. It is necessary to do the experiment about ten times in order to get the average of the makespan and energy values. In order to carry out the experiment, m random tasks were distributed over n virtual computers. A consistent distribution of work size and the amount of time required for execution is observed. It has been shown that the Gaussian technique works

better than the traditional PSO, and it also enhances the ability of the PSO to converge and converge more quickly. Given the multi-objective nature of our issue, using Gaussian with MOPSO and BFO yields superior energy efficiency and shorter makespan time. For mobile cloud computing to solve the offloading issue, both elements must be present[23].

Table 1 Simulation parameters that were taken into consideration

Parameters for BFO and PSO	**Value Used**
No_of_bacteria (*Np*)	20
No_of_chemotactics (*Nc*)	10
swim_length (*S l*)	4
No_of_reproductions (*Nr*)	4
No_of_dispersals (*Ne*)	2
step_size (C)	1.45
probability_dispersal (*P dispersal*)	0.25
d_attractant (*dattract*)	0.1
w_attractant (*wattract*)	0.2
h_repellant (*hattract*)	0.1
w_repellant (*wattract*)	10
PSO Swarm size	20
Self-recognition coefficient	1
Social coefficient	2
Inertial weight	0.5

6.2 Results and Discussion

The different job execution durations are shown in Table 2, where it is evident that the GMOPSO-BFO technique outperformed the other algorithms. The suggested plan keeps the lowest makespan even when the number of processes on the virtual machine rises. Compared to MOPSO, BFO, and MOPSO-BFO, the suggested system has a shorter makespan for the different range of jobs from 100 to 1000.

Table 2 Execution time of the task in different techniques

TASK	MOPSO	BFO	PSO-BFO	GMOPSO-BFO
100	41.47	38.66	37.65	37.18
200	155	151.81	155.05	145.25
300	345.4	335.53	335	327.38
400	594.85	597.26	594.85	567.8
500	926.43	916.66	913.98	878.55
600	1332.22	1333.36	1324.33	1284.5
700	1813.15	1885.31	1803.56	1750.51
800	2349.87	2390.75	2310.6	2316.66
900	2934.87	3034.93	2924.65	2916.73
1000	3743.78	3692.85	3655.96	3618.93

This paper calculates the energy consumption of the suggested GMOPSO-BFO approach and compares it with MOPSO[21], BFO[24,25], and MOPSO-BFO methods. In this experiment, there are five virtual computers with between one hundred and one thousand jobs. The experiment's goal was to find out how much energy each approach on virtual machines used. The energy usage is measured in joules per minute.

Table 3 Energy consumption of the task in different techniques

TASK	MOPSO	BFO	PSO-BFO	GMOPSO-BFO
100	1.05	1.05	1.03	1.03
200	2.15	2.15	2.15	2.14
300	3.14	3.12	3.16	3.11
400	4.15	4.13	4.15	4.12
500	5.18	5.19	5.17	5.18
600	6.33	6.33	6.3	6.18
700	7.45	7.46	7.5	7.42
800	8.49	8.47	8.46	8.45
900	9.6	9.62	9.61	9.59
1000	10.72	10.75	10.71	10.66

Table 3 presents the many tasks that have been performed on the virtual machine, as well as the GMOPSO-BFO approach, which has used a lower amount of energy measured in joules compared to the other algorithms. It has been observed that the amount of energy that the machine consumes rises as the number of tasks increases from one hundred to one thousand. In contrast to the previous approach, the schemes that were recommended perform much better. This method has the potential to lower the amount of energy that the virtual machine consumes [26,27]. The results of the experiments show that the GMOPSO-BFO system that was recommended is more effective in terms of the amount of energy it consumes and the amount of time it takes to finish.

Conclusion

Within the scope of this study, we provided a one-of-a-kind multi-objective task scheduling approach that was developed with Mobile Cloud Computing (MCC) environments in mind. The approach that has been presented aims to strike a compromise between a number of conflicting objectives, including employment completion time, energy consumption, and resource use. We framed the task scheduling issue as a multi-objective optimization problem and came up with a solution that was based on heuristics in order to properly divide work among the mobile and cloud resources that were available. Experimental assessments indicated that the suggested scheme is both successful and superior to current techniques in terms of establishing better trade-offs among competing goals. This was proved by the proposal's ability to achieve better results. To further improve the speed and scalability of task scheduling in MCC systems, future research objectives include researching sophisticated optimization methods and machine learning algorithms. This will hopefully lead to additional advancements in the field. In the publication, a hybrid scheduling method was presented. This method was based on Gaussian-based multi-objective particle swarm optimization (GMOPSO) and Bacterial foraging optimization (BFO). When we use the BFO, we are attempting to improvise the best solution that is available locally, but the GMOPSO gives us the greatest option that is available globally. Within this chapter, both the work methodology and the detailed design approach of the recommended scheduling system are provided to the reader. In addition, the chapter discusses the assessment outcomes in comparison to the works that have already been presented.

References:

[1] M. Smit, M. Shtern, B. Simmons, and M. Litoiu, “Partitioning Applications for Hybrid and Federated Clouds,” 2012, [Online]. Available: http://www.mikesmit.com/wp-content/papercitedata/pdf/cascon2012.pdf%5Cnpapers2://publication/uuid/8BB68396-C5E5-475D833B-CC1C08B39FD8.

[2] C. Wang and Z. Li, “Parametric analysis for adaptive computation offloading,” in Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation - PLDI ’04, 2004, p. 119, doi: 10.1145/996841.996857.

[3] J. Niu, W. Song, and M. Atiquzzaman, “Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications,” J. Netw. Comput. Appl., vol. 37, pp. 334–347, Jan. 2014, doi: 10.1016/j.jnca.2013.03.007.

[4] T. Verbelen, T. Stevens, F. De Turck, and B. Dhoedt, “Graph partitioning algorithms for optimizing software deployment in mobile cloud computing,” Futur. Gener. Comput. Syst., vol. 29, no. 2, pp. 451–459, 2013, doi: 10.1016/j.future.2012.07.003.

[5] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, “Cuckoo: A computation offloading framework for smartphones,” Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST, vol. 76 LNICST, pp. 59–79, 2012, doi: 10.1007/978-3-642-29336-8_4

[6] Z. Liu, X. Zeng, W. Huang, J. Lin, X. Chen, and W. Guo, “Framework for contextaware computation offloading in mobile cloud computing,” Proc. - 15th Int. Symp. Parallel Distrib. Comput. ISPDC 2016, pp. 172–177, 2017, doi: 10.1109/ISPDC.2016.30.

[7] P. A. L. Rego, E. Cheong, E. F. Coutinho, F. A. M. Trinta, M. Z. Hasany, and J. N. D. Souza, “Decision Tree-Based Approaches for Handling Offloading Decisions and Performing Adaptive Monitoring in MCC Systems,” Proc. - 5th IEEE

Int. Conf. Mob. Cloud Comput. Serv. Eng. MobileCloud 2017, pp. 74–81, 2017, doi: 10.1109/MobileCloud.2017.19.

[8] Q. Qi et al., "Knowledge-Driven Service Offloading Decision for Vehicular Edge Computing: A Deep Reinforcement Learning Approach," IEEE Trans. Veh. Technol., vol. 68, no. 5, pp. 4192–4203, 2019, doi: 10.1109/TVT.2019.2894437.

[9] S. Misra, B. E. Wolfinger, M. P. Achuthananda, T. Chakraborty, S. N. Das, and S. Das, "Auction-Based Optimal Task Offloading in Mobile Cloud Computing," IEEE Syst. J., vol. 13, no. 3, pp. 2978–2985, Sep. 2019, doi: 10.1109/JSYST.2019.2898903.

[10] D. Bhattacharjee, A. Rao, C. Shah, M. Shah, and A. Helmy, "Empirical modeling of campus-wide pedestrian mobility: observations on the USC campus," in IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004, vol. 4, pp. 2887– 2891, doi: 10.1109/VETECF.2004.1400588.

[11] M. Nir, A. Matrawy, and M. St-Hilaire, "An energy optimizing scheduler for mobile cloud computing environments," Proc. - IEEE INFOCOM, pp. 404–409, 2014, doi: 10.1109/INFCOMW.2014.6849266.

[12] H. Shah-Mansouri, V. W. S. Wong, and R. Schober, "Joint Optimal Pricing and Task Scheduling in Mobile Cloud Computing Systems," IEEE Trans. Wirel. Commun., vol. 16, no. 8, pp. 5218–5232, 2017, doi: 10.1109/TWC.2017.2707084.

[13] C. Arun and K. Prabu, "A multi-objective EBCO-TS algorithm for efficient task scheduling in mobile cloud computing," Int. J. Netw. Virtual Organ., vol. 22, no. 4, pp. 366–386, 2020, doi: 10.1504/IJNVO.2020.107570.

[14] M. Garg and R. Nath, "Autoregressive dragonfly optimization for multiobjective task scheduling (ado-mts) in mobile cloud computing," J. Eng. Res., vol. 8, no. 3, pp. 71–90, 2020, doi: 10.36909/JER.V8I3.7643.

[15] B. Hendrickson and T. G. Kolda, "Graph partitioning models for parallel computing," Parallel Comput., vol. 26, no. 12, pp. 1519–1534, 2000, doi: 10.1016/S0167-8191(00)00048-X.

[16] Seungjun Yang et al., "Fast dynamic execution offloading for efficient mobile cloud computing," in 2013 IEEE International Conference on Pervasive Computing and Communications (PerCom), Mar. 2013, pp. 20–28, doi: 10.1109/PerCom.2013.6526710.

[17] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," Appl. Comput. Informatics, vol. 14, no. 1, pp. 1– 16, Jan. 2018, doi: 10.1016/j.aci.2016.11.002.

[18] P. A. L. Rego, P. B. Costa, E. F. Coutinho, L. S. Rocha, F. A. M. Trinta, and J. N. de Souza, "Performing computation offloading on multiple platforms," Comput. Commun., vol. 105, pp. 1–13, Jun. 2017, doi: 10.1016/j.comcom.2016.07.017.

[19] Ferrari, S. Giordano, and D. Puccinelli, "Reducing your local footprint with anyrun computing," Comput. Commun., vol. 81, pp. 1–11, May 2016, doi: 10.1016/j.comcom.2016.01.006

[20] S. Bermejo and J. Cabestany, "Adaptive soft k-nearest-neighbour classifiers," Pattern Recognit., vol. 33, no. 12, pp. 1999–2005, Dec. 2000, doi: 10.1016/S0031- 3203(99)00186-7.

[21] W. B. Claster, "Naïve Bayes Classifier," in Mathematics and Programming for Machine Learning with R, CRC Press, 2020, pp. 141–160.

[22] S. Menard, Applied Logistic Regression Analysis. 2455 Teller Road, Thousand Oaks California 91320 United States of America: SAGE Publications, Inc., 2002.

[23] O. Sagi and L. Rokach, "Ensemble learning: A survey," Wiley Interdiscip. Rev. Data Min. Knowl. Discov., vol. 8, no. 4, Jul. 2018, doi: 10.1002/widm.1249.

[24] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Futur. Gener. Comput. Syst., vol. 25, no. 6, pp. 599–616, 2009, doi: 10.1016/j.future.2008.12.001.

[25] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, “A survey of mobile cloud computing: architecture, applications, and approaches,” Wirel. Commun. Mob. Comput., vol. 13, no. 18, pp. 1587–1611, Dec. 2013, doi: 10.1002/wcm.1203.

[26] S. Singh and I. Chana, “A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges,” J. Grid Comput., vol. 14, no. 2, pp. 217–264, Jun. 2016, doi: 10.1007/s10723-015-9359-2.

[27] E. E. Marinelli, “Hyrax : Cloud Computing on Mobile Devices,” vol. 0389, no. September, 2009.