ISSN 2007-9737

A Multi-Objective Task Scheduling Scheme GMOPSO-BFO in Mobile Cloud Computing

Robin Prakash Mathur[1], Manmohan Sharma[2]

[1] School of Computer Science and Engineering,
School of Computer Applications, Lovely Professional University, Punjab,
India

[2] Lovely Professional University, Punjab,
India

mathur.robin@gmail.com

Abstract. Mobile cloud computing is currently an encouraging field in the cyber-physical world. It is an amalgamation of mobile computing and cloud computing. Computational offloading is one feature in the mobile cloud application that offloads the task to the cloud server, processes it, and gets the results back on the mobile device. During offload, the job needs to be queued on the cloud servers and allocated to the virtual machines. Task scheduling is an important step where the mobile task is assigned to the servers and processed somehow. In the overall offloading process, energy conservation is a significant concern. The scheduling problem involves mapping the offloaded task to the cloud server while satisfying the energy and time constraints. This paper proposes a hybrid scheduling scheme based on Gaussian-based multi-objective particle swarm optimization(GMOPSO) and bacterial foraging optimization(BFO). This scheme performs better when compared to other variants of PSO in terms of makespan and energy efficiency.

Keywords. Computational offloading, mobile cloud computing, MOPSO, bacteria foraging optimization, energy consumption, makespan.

1 Introduction

The recent studies by Ericsson (Ericsson Mobility Report, n.d.) show that mobile users will become 25 billion by 2025. The rise of internet connectivity and low-cost mobile devices are some of the reasons for the increasing number of mobile users. Still, battery constraints is existing as one of the mobile device's limitations. Today's mobile applications exhaust the device battery in a fast manner and also require higher computational requirements. The solution to these problems can be handled by computational offloading of mobile edge computing.

Mobile edge computing is an amalgamation of mobile computing and edge computing. It is a closer infrastructure to the user device as compared to cloud computing. The properties like small-scale data centers, location nearby LTE or Wi-Fi, low latency, dense deployment by telecom vendors, and lower congestion make it a better option for mobile task offloading applications.

Computation offloading is the technique inside mobile edge computing where an application is partitioned upon local and remote execution based on some criteria. Fig 1. depicts the offloading process where an application is partitioned, and based upon some measures, the decision has been taken to offload the task or execute it locally. Those tasks which are identified to be performed on an edge server are offloaded on it.

These jobs reach the cloud server and get scheduled by some scheduling technique. Task scheduling on the cloud server is one of the prime tasks in mobile cloud computing.

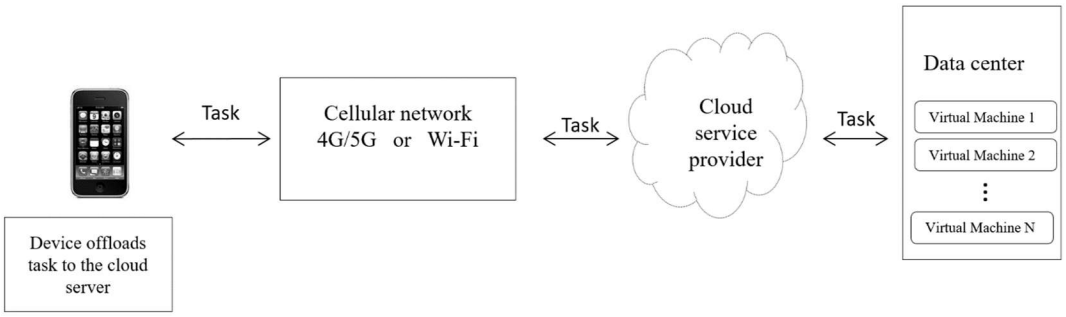

Fig. 1. Offloading process in the mobile computing environment

The virtual machines (VM) need to be allocated to the task's execution by the cloud service provider. Major thirst has been given to research in the field of mobile computing by a framework like Chroma(RPC) (Balan et al., 2003), Cuckoo(RMI) (Kemp et al., 2012), spectra(Flinn et al., n.d.), MAUI (Cuervo et al., 2010), Mobicloud (Huang et al., 2010), and Clonecloud (Chun et al., 2011). These are popular frameworks in this cloud computing domain that empower the concept of offloading the task to the cloud server either by task partitioning or considering a complete application for offloading purposes.

Various studies have been done in the past, trying to achieve optimization in different objective functions like makespan, energy, quality of service(QoS), load balancing, and cost. The problem of task scheduling has much scope for optimization since of its NP-hard nature.

The mobile application consists of many computational tasks represented as nodes and dependency among these nodes is defined as a cloud. Resources are required in the cloud servers for the execution of these offloaded computational tasks. The availability of these resources needs to be assured by the cloud service providers, and also, the pricing of services may vary from country to country.

The work aims to propose a hybrid scheduling technique based on Gaussian-based multi-objective particle swarm optimization (GMOPSO) and Bacterial foraging optimization (BFO). The GMOPSO provides us with the global best solution, whereas using the BFO, the local best solution is tried to be improvised. The contribution can be summarized as follows.

a) Minimize the energy consumption and makespan of the scheduling process.
b) Simulation and performance evaluations of the proposed algorithm with existing approaches.

In Section 2, related literature has been reviewed. Section 3 describes the methodology of the work. The detailed design approach of the suggested system is presented in Section 4. Section 5 offers the evaluation results compared with existing works, and the conclusion and future directions are shown in the last in section 6.

2 Literature Review

This section provides the work done so far in the field of scheduling in mobile cloud and cloud computing. Once the task has been offloaded to a virtual machine, its execution plan or schedule is another challenge.

The scheduling algorithm must be optimally designed so that the task's timely execution can be achieved and starvation or deadlock-like conditions can be avoided. Eom et al. (2013) focused on scheduling offloading and applying machine learning-based techniques to optimize the offloading process.

Their study focused on nineteen different machine learning algorithms and four workloads. Zhang et al. (2016) have proposed joint resource scheduling and code partitioning for effectively allocating cloudlet to multiple cloud users. They have proposed a code partitioning algorithm based on the call tree. Hsu Mon Kyi & Thinn Thu Naing

(2011) have proposed an algorithm for scheduling and resource allocation of virtual resources and virtual machines named the Efficient Virtual Machines Scheduling Algorithm (EVMSA). The stochastic Markov model is used to analyze the performance of the scheduling algorithm. Eucalyptus architecture is introduced as a system model. The resource allocation decision model is based on the continuous Markov chain model. Jagannathan & Modiano (2013) have presented a mathematical model of the buffer overflow in parallel queues.

The study shows that the longest queue first scheduling policy has a superior queue overflow performance than queue blind policies. Several lemmas are presented in support of the theory presented in the paper. The researcher has assumed the system consists of N parallel queues served by a single server. Time is slotted, and the server processes only one queue. Wang et al. (2013) presented the weighted round-robin scheduling algorithm for task scheduling in the Hadoop framework.

Table 1 presents the various task scheduling schemes, specifically in the mobile cloud computing framework. In the paper Wei et al. (2013), the authors have proposed the extended cloudlet approach for supporting local mobile cloud.

They have presented a hybrid PSO approach and optimized the profit and energy consumption during scheduling. The authors in the paper Nir et al. (2014) have presented a task scheduler model that optimizes mobile cloud computing's energy function. In the paper Lin et al. (2015), the authors proposed a scheduling scheme based on dynamic voltage and frequency scaling and optimized the application makespan and reduce energy consumption.

3 Task Offloading & Scheduling in Mobile Cloud Computing

The mobile task offloading model consists of two ways to execute the task, i.e., either to offload the task on the cloud server or to execute the task locally on the mobile phone. After the initial task partitioning phase, the decision of offloading is made by the decision engine by gathering various device and network parameters through the profiling process. Now, through a cellular network or Wi-Fi network, the task reaches the cloud server. The objective of offloading is to transfer the computation to the resourceful server at a distant place to improve the device's performance and save energy. Taking the offload decision to a remote server is not always mandatory but depends on the various parameters affecting the device's performance.

In some scenarios, partial offloading is also performed. One part of the application task is processed on a mobile device, and the other is offloaded to the surrogate or cloud server. The task's computation time depends on the computation amount required and the mobile device's processing speed. In a scenario, let's assume the job is divided into two partitions where the first partition executes locally and the second partition runs on a remote server.

For local execution, let C_{T_LOCAL} be the computation time required on the local device, C_{A_LOCAL} be the computation amount, and P_{S_LOCAL} be the mobile device's processing speed. The relationship among these values will be:

$$C_{T\_LOCAL} = C_{A\_LOCAL} / \quad P_{S\_LOCAL}. \tag{1}$$

For remote execution, the second partition is executed on the cloud/edge server. Let $B_{AVAILABLE}$ be the available bandwidth in the device and the amount of data to be transferred be $D_{AMOUNT\_OF\_DATA}$. The time taken to transfer the data to/from the server will be C_{T_REMOTE} will be:

$$C_{T\_REMOTE} = D_{AMOUNT\_OF\_DATA} / B_{AVAILABLE} + \text{Cloud processing time}. \tag{2}$$

The total time C_{T_TOTAL} taken to execute the application both locally and remotely will be a summation of the above two equations, which is:

$$C_{T\_TOTAL} = C_{T\_LOCAL} + C_{T\_REMOTE}. \tag{3}$$

3.1 Cloud Model

When the task is offloaded from the mobile device to the cloud server, it reaches the cloud service provider's server. The cloud service provider manages all information about the task that is approached for processing.

Table 1. Various scheduling schemes in MEC / MCC environment in the recent past

Techniques and Work Done	Year	Type of Problem	Objectives function	Framework	Environment
HACAS (Wei et al., 2013)	2013	Application scheduling	Profit and Energy consumption	MCC	Simulation
TSPCCE (Nir et al., 2014)	2014	Task scheduling	Energy	MCC	IBM's linear programming solver
MCC task scheduling algorithm(X. Lin et al., 2015)	2014	Task scheduling with DVFS	Energy and Time	MCC	MATLAB
LARAC algorithm (W. Zhang et al., 2015)	2015	Task scheduling with DVFS	Energy and Time Deadline	MCC	Simulation
eDors (Guo et al., 2016)	2016	Dynamic scheduling and energy-efficient offloading	Energy and completion time	MCC	Simulation
MCF-DF (Lin Wang et al., 2016)	2016	Task admission and scheduling	Admission rate and execution cost	MEC	Python
HCOA(T. Wang et al., 2018)	2017	Task offloading and scheduling	Energy	MCC	Simulation
CMSACO (Shah-Mansouri et al., 2017)	2017	Multi-Task offloading	Profit and completion time	MCC	Simulation
TSRA(Zhao et al., 2017)	2017	Resource allocation and scheduling	Delay	MEC	Simulation
COPE (J. Zhang et al., 2018)	2017	Task scheduling	Energy, Price of Cloud service provider, Delay	MCC	Thinkair based simulation
DAA (L. Lin et al., 2018)	2018	Task scheduling	Makespan	MEC	Simulation

GABTS (Tang et al., 2018)	2018	Task offloading and scheduling	Energy, response time, deadline, and cost	MCC	C++
OAOA (Jiang et al., 2019)	2019	Stochastic approach for task scheduling	Energy and QoS	MCC	Simulation
Application-aware (Oo & Ko, 2019)	2019	Task Scheduling	Latency	MEC	iFogSim
MWSM (Tian et al., 2019)	2019	Workflow scheduling	Latency, Energy, and Cost	MCC	Simulation
RCTSPO (Chen et al., 2020)	2020	Task scheduling	Makespan, Reliability, and Load	MEC	Cloudsim
EBCO-TS (Arun & Prabu, 2020)	2020	Task scheduling	Makespan and energy	MCC	Cloudsim
ADO-MTS (Garg & Nath, 2020)	2020	Task scheduling	Makespan, Resource utilization, and Energy	MCC	Cloudsim

The Datacenter Broker policy (Singh & Chana, 2016) helps the cloudlets (task) to assign virtual machines.

The data center policy must be appropriate for the minimum execution time of the cloudlet. Similar to web applications, a mobile application consists of different tasks.

These tasks can be represented as a directed acyclic graph (DAG). While the application's independent task can be executed simultaneously in multiple virtual machines, the dependent job needs to be synchronized as per their precedence order.

3.2 Scheduling of Offloaded Task

When speaking about task scheduling, achieving a minimum makespan is considered an NP-hard problem. Most recent studies have focused on the cloud resources to the various cloudlets to optimize energy and execution time parameters.

In this work, the particular task's execution time depends on the task size and the virtual machine's property. Following are the basic definitions regarding mobile task scheduling:

a) Consider a set of n virtual machines as V = {V_1, V_2, V_3…, V_n}

b) A task of the application tasks T = {T_1, T_2…, T_x}

c) E is the set of connections between any two tasks, T_i and T_j.

d) Collection of physical machines (PMs) in the data center = (PM_1, PM_2, PM_3..., PM_n)

It is assumed in the work that the cloud service provider has a sufficient number of computational resources. The V number of virtual machines are deployed on the physical machines, and different virtual machines have a variety of processing units (CPU), random access memory (RAM), and networking capabilities.

The data center brokers monitor all available resources and assign the machine to the task once approached. All jobs requiring processing resources need to stand in a queue, and based on the task scheduling scheme, tasks are planned to execute on the machine.

3.3 Framework for Task Scheduling

An approach has been proposed on a Gaussian multi-objective hybrid scheduling scheme based on particle swarm optimization (PSO) and bacterial foraging optimization (BFO). Energy and

makespan are considered objective functions for the study. A task offloading scheme is based on optimizing the multi-objective function, where minimizing both functions is the approach's actual goal.

Makespan is defined as the time required for the processing of the task CPU and its transmission time. The makespan of a task on the virtual machine is calculated considering the computing power of the VM and the size of the task. It can be defined by the following equation:

$$\text{Makespan(T)} = \text{size of the task} / \text{computational power.} \quad (4)$$

Two factors calculate energy cost: virtual machine usage charges, which are usually different for cloud service providers, and calculated on a second basis. The other is the execution time of the task. It can be defined by the following equation:

$$\text{Energy Cost} = \text{execution time X virtual machine usage charge.} \quad (5)$$

3.4 Objective Functions

This section defines the objective functions considered in the work

Objective 1: The first aim of the function is to minimize the makespan of the mobile task.

Objective 2: The second aim of the function is to minimize the energy cost of the mobile task.

4 Proposed Approach for Task Scheduling

The proposed approach (GMOPSO-BFO) is based on a hybrid approach of particle swarm optimization (PSO) and Bacteria foraging optimization (BFO). The PSO approach works excellently in searching the solution globally, whereas the BFO works optimally with local search capabilities.

The combined approach of these two techniques generates an optimal solution globally and locally in search capability and higher convergence time.

4.1 Bacteria Foraging Optimization

The bacteria foraging method is a natural selection method in which microorganisms like bacteria tend to search or forage food to survive in the E-coli (intestine) of the human body (Passino 2002). The primary strategy of bacteria to survive is by locating the nutrients, handling them, and ingesting the food to get the energy to live and reproduce. Those bacteria which do not successfully forage the nutrient typically get eliminated from the system. It follows the concept of survival of the fittest.

This evolutionary concept made the scientist fascinated and motivated them to use it as an optimization process. Most of the optimization processes can be performed with such an evolutionary approach. The main aim of the bacteria is to maximize the energy attained during foraging per unit of time. It depends on certain factors like prey density in the environment and characteristics of the environment. It also depends on the sensing and cognitive capabilities of the bacteria.

The E. Coli bacteria have a cell structure having various biological features like nucleoids, ribosomes, cytoplasm, pilus, and plasma membrane. As these attributes do normal cell processes, another critical feature, i.e., flagellum, helps bacteria propagate or move in different directions. Chemotaxis is the process of movement of the organism from its position in the presence of some chemical attractants and repellants.

With the help of flagella, there are two possible movements, i.e., either its moves clockwise or tumble, and the other is counterclockwise or swim. Fig.2 depicts the movement of bacteria like tumbling and swimming in the E. Coli.

In a favorable condition of the environment where sufficient nutrients are available and the non-acidic and non-alkaline nature of the intestine, it swims and the opposite of it. It tumbles typically, which is changing the direction of the swim.

The other significant process related to bacteria is swarming, where bacteria release some attractants to swarm together, searching for food. If the attractants are released high and deep, there are chances that different bacteria explore food together; otherwise, they go alone in the reverse situation.

During the reproduction process, the bacteria get split into two parts to increase their population. Bacteria reproduce based on the nutrient available in the bacteria or the fitness function. The bacteria also go through the elimination and dispersal phase in their lifetime due to their local environment.

Sometimes, the condition to survive gets reduced when the sudden rise in heat or nutrients is finished. In terms of computing, to avoid trapping in the local optima, the elimination and dispersal process is used.

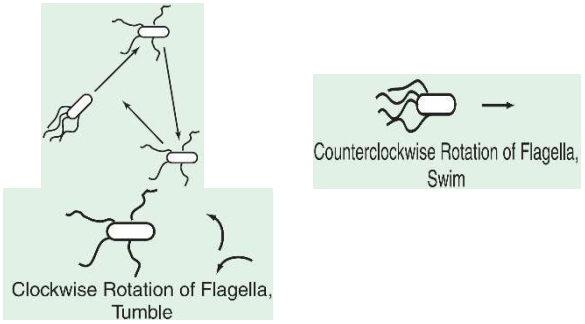

Fig. 2. Chemotaxis process of the bacteria

4. 2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a nature-inspired algorithm (Coello Coello & Lechuga, n.d.) (Krohling, n.d.) based on social behavior and a flock of birds' dynamic movement. A group of birds known as a swarm moves together, searching for food in a particular direction and at different velocities. Each bird or particle looks for food and is usually followed by other birds.

These birds communicate with each other during their search and typically follow each other closer to the food. The closeness from the food is calculated as a fitness value after a periodic interval of time. Each bird in the swarm is represented as a particle in multidimensional space with a certain velocity and position.

Each particle keeps two things in its memory, i.e., its own best position pbest, and other is the global best position of gbest of their group. In the standard PSO, the velocity of the particle is updated with the equation:

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

The updated version of PSO, which improve the convergence rate, was a constriction factor where the velocity vector as:

$$v_i^{(k+1)} = \chi\begin{bmatrix} v_i^{(k)} + c_1\xi_1(b_i^{(k)} - x_i^{(k)}) + c_2\xi_2 \\ (y^{(k)} - x_i^{(k)}) \end{bmatrix}, \quad (7)$$

$$x_i^{(k+1)} = x_i^{(k)} + v_i^{(k+1)}, \quad (8)$$

where χ is a constriction factor in the above equation and $x_i^{(k)}$ is the ith particle's position at step k, $v_i^{(k+1)}$ is its velocity, $b_i^{(k)}$ is the best position visited by the i[th] particle, $y^{(k)}$ is the overall best position ever visited. It has been observed that after incorporating the Gaussian density function in the above equation, the results come better in terms of the global solution. The updated velocity equation will be:

$$v_i^{(k+1)} = \left[\| randn | (b_i^{(k)} - x_i^{(k)}) + Randn | (y^{(k)} - x_i^{(k)}) \right. \quad (9)$$

where the randn and Randn are based on the Gaussian density function's absolute value.

The Gaussian random density function is represented by:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \quad (10)$$

Pseudocode of 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 is represented as $\theta^i(j,k,l)$

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

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.

for all bacteria in the list:

Loop elimination-dispersal step

Loop reproduction step

Table 2. Parameters considered in the simulation

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

Loop Chemotaxis step

go for chemotactic steps using (a) and (b), respectively

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:

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

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

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

Update J_{last}

For the reproduction phase: calculate the fitness function using:

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

Sort in ascending order the bacteria and chemotactic parameters If ($k < N_r$), perform the reproduction step again till $k = N_r$

For elimination and dispersal:

for each bacteria,

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

do elimination and dispersal till $l = N_e$.

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

Update p_i, best, and g_i, best upon meeting the condition:

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

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

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

Update the position of each bacteria (particle) after every iteration by the formula:

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

Check p_i, which should exist within the range.

Repeat step reproduction and PSO until convergence is achieved.

After the stopping criteria are met, the value of g_{best} and $f(g_{best})$ must be recorded. End.

5 Results and Discussion

The proposed approach has been developed in the language Python in the window 10 environment on Intel (R) Core (TM) i5, 1.80 GHz, CPU 8 GB. Various parameters considered during the simulation of the proposed technique have been presented in Table 2.

In evaluating the proposed method, five virtual machines are considered, and a collection of tasks is assumed between 100 and 1000. The results are compared with the existing work on MOPSO (Alkayal et al., 2016) and BFO (Rajni & Chana, 2013) regarding the energy efficiency and makespan of the task execution.

The proposed scheme is based Gaussian swarm approach implemented in MOPSO along with the BFO. The experiment has been performed by considering the number of bacteria (*Np*) as 20 and No_of_chemotactics (*Nc*) as 10. In the same way, the initial size of PSO is considered as 20 in the experiment. The experiment runs iteratively about ten times to find the average of makespan and energy values. The experiment has been

Table 3. Execution time of the task in different techniques

		MOPSO	BFO	MOPSO-BFO	GMOPSO-BFO
Makespan per no. of task	**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

Table 4. Energy consumption of the task in different techniques

		MOPSO	BFO	MOPSO-BFO	GMOPSO-BFO
Energy consumed per no. of task	**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

performed by considering *m* random task to *n* virtual machine.

The task size and required execution time are uniformly distributed. It has been found that the Gaussian scheme has outperformed the standard PSO and increased the convergence ability of PSO.

Since our problem is multi-objective, when Gaussian is implemented with MOPSO along with BFO, it gives better results in energy efficacy and reduced makespan time. Both factors are required for the offloading problem in mobile cloud computing. Table 3 presents the various task execution times, and it can be seen that the GMOPSO-BFO approach has performed better than the other algorithms.

As the number of tasks increases on the virtual machine, the proposed scheme maintains the lowest makespan. The proposed scheme has less makespan for the various range of task from 100 to 1000 compared to MOPSO, BFO, and MOPSO- BFO.

In this work, the energy consumption is calculated for the proposed GMOPSO-BFO technique and compared with methods like MOPSO, BFO, and MOPSO-BFO. In this experiment, the number of virtual machines is

considered 5, and the number of tasks ranges from 100 to 1000.

The experiment aimed to determine the energy consumption of the various techniques on the virtual machines.

The unit of energy consumption is considered as joules/minute. Table 4 presents the various tasks on the virtual machine and GMOPSO-BFO approach, which has consumed less energy in joules than the other algorithms. It has been observed that when the number of tasks increases from 100 to 1000, the machine's energy consumption also increases.

The proposed schemes perform better as compared to the other algorithm. The proposed scheme can save energy consumption in the virtual machine. It is clear from the experimental results that the proposed scheme GMOPSO-BFO performs better in completion time and energy consumption.

6 Conclusions

This paper presents a hybrid scheduling approach based on the Gaussian multi-objective particle swarm optimization and bacteria foraging optimization. Both makespan and energy consumption are essential factors in the offloading method of MCC. The proposed scheme performs better in makespan and energy consumption.

The results are compared with the MOPSO, BFO, and hybrid MOPSO-BFO. The scheme leverages the global optima of GMOPSO and the local optima by BFO. In the future, a scheduling scheme will be developed based on other optimization parameters like a load on the servers, scalability, latency, and resource utilization.

References

1. **Alkayal, E. S., Jennings, N. R., Abulkhair, M. F. (2016).** Efficient Task scheduling multi-objective particle swarm optimization in cloud computing. IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops), pp. 17–24. DOI: 10.1109/LCN.2016.024.
2. **Arun, C., Prabu, K. (2020).** A multi-objective EBCO-TS algorithm for efficient task scheduling in mobile cloud computing. International Journal of Networking and Virtual Organisations, Vol. 22, No. 4, pp. 366–386. DOI: 10.1504/IJNVO.2020.107570.
3. **Balan, R. K., Satyanarayanan, M., Park, S. Y., Okoshi, T. (2003)**. Tactics-based remote execution for mobile computing. Proceedings of the 1st International Conference on Mobile Systems, Applications and Services MobiSys '03, pp. 273–286. DOI:10.1145/1066116.1066125.
4. **Chen, L., Guo, K., Fan, G., Wang, C., Song, S. (2020).** Resource constrained profit optimization method for task scheduling in edge cloud. IEEE Access, Vol. 8, pp. 118638-118652. DOI:10.1109/ACCESS.2020.3000985.
5. **Chun, B. G., Ihm, S., Maniatis, P., Naik, M., Patti, A. (2011).** CloneCloud: Elastic execution between mobile device and cloud. Proceedings of the Sixth Conference on Computer Systems, pp. 301–314, DOI: 10.1145/1966445.1966473.
6. **Coello Coello, C. A., Lechuga, M. S. (2002).** MOPSO: A proposal for multiple objective particle swarm optimization. Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, Vol. 2, pp. 1051–1056. DOI: 10.1109/CEC.2002.1004388.
7. **Cuervo, E., Balasubramanian, A., Cho, D., Wolman, A., Saroiu, S., Chandra, R., Bahl, P. (2010).** MAUI: Making smartphones last longer with code offload. Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services MobiSys '10, pp. 49–62. DOI: 10.1145/1814433.1814441.
8. **Eom, H., Juste, P. St., Figueiredo, R., Tickoo, O., Illikkal, R., Iyer, R. (2013).** Machine Learning-Based Runtime Scheduler for Mobile Offloading Framework. 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, pp. 17–25. DOI: 10.1109/UCC.2013.21.
9. **Flinn, J., SoYoung Park, Satyanarayanan, M. (2002)**. Balancing performance, energy, and quality in pervasive computing. Proceedings 22nd International Conference on

Distributed Computing Systems, pp. 217–226. DOI: 10.1109/ICDCS.2002.1022259.

10. **Garg, M., Nath, R. (2020).** Autoregressive dragonfly optimization for multi-objective task scheduling (ado-mts) in mobile cloud computing. Journal of Engineering Research (Kuwait), Vol. 8, No. 3, pp. 71–90. DOI: 10.36909/JER.V8I3.7643.
11. **Guo, S., Xiao, B., Yang, Y., Yang, Y. (2016).** Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. The 35th Annual IEEE International Conference on Computer Communications, pp. 1–9, DOI: 10.1109/INFOCOM.2016.7524497.
12. **Kyi, H. M., Naing, T. T. (2011).** Stochastic markov model approach for efficient virtual machines scheduling on private cloud. International Journal on Cloud Computing: Services and Architecture, Vol. 1, No. 3, pp. 1–13. DOI: 10.5121/ijccsa.2011.1301.
13. **Huang, D., Zhang, X., Kang, M., Luo, J. (2010).** MobiCloud: Building secure cloud framework for mobile computing and communication. Fifth IEEE International Symposium on Service Oriented System. DOI: 10.1109/SOSE.2010.20.
14. **Jagannathan, K., Modiano, E. (2013).** The impact of queue length information on buffer overflow in parallel queues. IEEE Transactions on Information Theory, Vol. 59, No. 10, pp. 6393–6404. DOI: 10.1109/TIT.2013.2268926.
15. **Jiang, Q., Leung, V. C. M., Tang, H., Xi, H. S. (2019).** Adaptive scheduling of stochastic task sequence for energy-efficient mobile cloud computing. IEEE Systems Journal, Vol. 13, No. 3, pp. 3022–3025. DOI: 10.1109/JSYST.2019.2922436.
16. **Passino, K. M. (2002).** Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems, Vol. 22, No. 3, pp. 52–67. DOI: 10.1109/MCS.2002.1004010.
17. **Kemp, R., Palmer, N., Kielmann, T., Bal, H. (2012).** Cuckoo: A Computation offloading framework for smartphones. In: Gris, M., Yang, G. (eds) Mobile Computing, Applications, and Services, MobiCASE 2010, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol 76,DOI: 10.1007/978-3-642-29336-8_4.
18. **Krohling, R. A. (2004).** Gaussian swarm: A novel particle swarm optimization algorithm. IEEE Conference on Cybernetics and Intelligent Systems, Vol. 1, pp. 372–376. DOI: 10.1109/ICCIS.2004.1460443.
19. **Lin, L., Li, P., Xiong, J., Lin, M. (2018).** Distributed and application-aware task scheduling in edge-clouds. 2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), pp. 165–170. DOI: 10.1109/MSN.2018.000-1.
20. **Lin Wang, Jiao, L., Kliazovich, D., Bouvry, P. (2016).** Reconciling task assignment and scheduling in mobile edge clouds. IEEE 24th International Conference on Network Protocols (ICNP), pp. 1–6. DOI: 10.1109/ICNP.2016.7785317.
21. **Lin, X., Wang, Y., Xie, Q., Pedram, M. (2015).** Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. IEEE Transactions on Services Computing, Vol. 8, No. 2, pp. 175–186. DOI: 10.1109/TSC.2014.2381227.
22. **Nir, M., Matrawy, A., St-Hilaire, M. (2014).** An energy optimizing scheduler for mobile cloud computing environments. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 404–409. DOI: 10.1109/INFCOMW.2014.6849266.
23. **Oo, T., Ko, Y. B. (2019).** Application-aware task scheduling in heterogeneous edge cloud. International Conference on Information and Communication Technology Convergence (ICTC), pp. 1316–1320. DOI: 10.1109/ICTC46691.2019.8939927.
24. **Rajni, Chana, I. (2013).** Bacterial foraging based hyper-heuristic for resource scheduling in grid computing. Future Generation Computer Systems, Vol. 29, No. 3, pp. 751–762. DOI: 10.1016/j.future.2012.09.005.
25. **Shah-Mansouri, H., Wong, V. W. S., Schober, R. (2017).** Joint optimal pricing and task scheduling in mobile cloud computing systems. IEEE Transactions on Wireless

Communications, Vol. 16, No. 8, pp. 5218–5232. DOI: 10.1109/TWC.2017.2707084.

26. **Singh, S., Chana, I. (2016).** A survey on resource scheduling in cloud computing: issues and challenges. Journal of Grid Computing, Vol. 14, No. 2, pp. 217–264. DOI: 10.1007/s10723-015-9359-2.
27. **Tang, C., Wei, X., Xiao, S., Chen, W., Fang, W., Zhang, W., Hao, M. (2018).** A mobile cloud based scheduling strategy for industrial internet of things. IEEE Access, Vol. 6, pp. 7262–7275. DOI:10.1109/ACCESS.2018.2799548.
28. **Tian, W., Gu, R., Feng, R., Liu, X., Fu, S. (2019).** A QoS-Aware workflow scheduling method for cloudlet-based mobile cloud computing. Proceedings EEE International Congress on Cybermatics: 12th IEEE International Conference on Internet of Things, 15th IEEE International Conference on Green Computing and Communications, 12th IEEE International Conference on Cyber, Physical and So, pp. 164–169. DOI:0.1109/iThings/GreenCom/CPSCom/SmartData.2019.00048.
29. **Wang, D., Chen, J., Zhao, W. (2013).** A task scheduling algorithm for Hadoop platform. Journal of Computers, Vol. 8, No. 4. DOI: 10.4304/jcp.8.4.929-936.
30. **Wang, T., Wei, X., Tang, C., Fan, J. (2018).** Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints. Peer-to-Peer Networking and Applications, Vol. 11, No. 4, pp. 793–807. DOI: 10.1007/s12083-017-0561-9.
31. **Wei, X., Fan, J., Lu, Z., Ding, K. (2013).** Application scheduling in mobile cloud computing with load balancing. Journal of Applied Mathematics, Vol. 2013. DOI: 10.1155/2013/409539.
32. **Yuan Zhang, Jinyao Yan, Xiaoming Fu. (2016).** Reservation-based resource scheduling and code partition in mobile cloud computing. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 962–967. DOI: 10.1109/INFCOMW.2016.7562219.
33. **Zhang, J., Zhou, Z., Li, S., Gan, L., Zhang, X., Qi, L., Xu, X., Dou, W. (2018).** Hybrid computation offloading for smart home automation in mobile cloud computing. Personal and Ubiquitous Computing, Vol. 22, No. 1, pp. 121–134. DOI: 10.1007/s00779-017-1095-0.
34. **Zhang, W., Wen, Y., Wu, D. O. (2015).** Collaborative task execution in mobile cloud computing under a stochastic wireless channel. IEEE Transactions on Wireless Communications, Vol. 14, No. 1, pp. 81–93. DOI: 10.1109/TWC.2014.2331051.
35. **Zhao, T., Zhou, S., Guo, X., Niu, Z. (2017).** Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing. IEEE International Conference on Communications (ICC), pp. 1–7. DOI: 10.1109/ICC.2017.7996858.

Article received on 28/04/2021; accepted on 17/04/2023.
Corresponding author is Robin Prakash Mathur.