Task Scheduling with Altered Grey Wolf Optimization (AGWO) in Mobile Cloud Computing using Cloudlet

J. Arockia Mary, and A. Aloysius

***Abstract*- Mobile devices can improve their battery life by offloading their tasks to a nearby cloudlet instead of executing tasks on the mobile device. Because mobile devices have low-speed processors, small-size memory, and limited battery. As the mobile devices are moving, they are connected and disconnected from the cloudlets. So, their tasks are offloaded to the new cloudlets and also migrated from one cloudlet to another until the tasks finish their execution. Scheduling these tasks in the cloudlet will reducethe tasks' execution time and the mobile device's power consumption using this proposed new method (AGWO). The GWO algorithm is modified to accept the inputs from a two-dimensional array instead of sequence inputs and search for the prey within the two-dimensional array instead of an unknown circle area. This method deals with the arrival time of the task, task size, and big task. The migration of the partially executed task dynamically to other VMs is also examined. This proposed method also reduces the average scheduling delay and increases the percentage of requests executed by the cloudlet than other variations of GWO and other research algorithms.**

***Index term*- Task scheduling, Grey wolf Optimization algorithm, Two-dimensional array input, Mobile cloud computing, Execution time, Energy**.

I. INTRODUCTION

Older mobile devices are lacking in battery power, processor speed, and memory size, but with the improvement in these resources, intelligent devices are upgraded to execute almost all types of tasks, including big tasks with big data. Some of the big tasks are weather monitoring, e-quake applications [1], Online shopping applications, live road applications, monitoring health condition applications, and user demand prediction applications. These applications run on mobile devices and consume more time for processing causing a reduction of the mobile device's battery energy quickly. By offloading and scheduling these applications in the cloud [2] and cloudlet [3], the mobile device can save energy and give the best experience to the user.

Manuscript received November 24, 2022; revised December 21, 2022. Date of publication March 22, 2023. Date of current version March 22, 2023. The associate editor prof. Claudia Canali has been coordinating the review of this manuscript and approved it for publication.

J. Arockia Mary is with the Department of Computer Applications, Holy Cross College (Affiliated to Bharthidasan University), Tiruchirappalli, Tamil Nadu, India (jarockia79@gmail.com).

A. Aloysiusis is with the Department of Computer Science, St. Joseph's College (Affiliated to Bharathidasan University), Tiruchirappalli, Tamil Nadu, India (aloysius1972@gmail.com).

Digital Object Identifier (DOI): 10.24138/jcomss-2022-0151

These applications are offloaded from mobile devices either to a distant cloud that requires many network hops and a lot of network resources. Instead of a distant cloud, a cloudlet near the access point requires one network hop is also used and reduces the requirement for more network resources. Many real-time mobile applications are offloaded from many mobile devices to the cloudlet. The cloudlet is a mini data center consisting of several virtual machines varying in memory size and processor speed. These virtual machines execute the offloaded tasks in the cloudlet. The numbers of offloaded tasks are more than the number of virtual machines resulting in many tasks waiting in a queue for processing, increasing the execution time of tasks. Without scheduling tasks, the tasks' waiting time increases. Major scheduling algorithms like FCFS (First Come First Serve), SJF (Shortest Job First), Round-Robin, and new scheduling schemes incorporate optimization algorithms like enhanced bee colony; genetic algorithms have been used to schedule the tasks. They reduce the cost, and power consumption of the cloudlet and increase the other quality service factors, namely throughput and execution time.

In this article, the Grey wolf optimization (GWO) algorithm is proposed [4] with alterations to accept inputs in a two-dimensional array instead of a one-dimensional array. GWO is a discrete optimization problem, NP-hard, and belongs to swarm intelligence techniques. GWO is a meta-heuristic algorithm [5] that generates random solutions from infinite solution space. GWO is a stochastic optimization algorithm [6] that uses three dominant search agents alpha, beta, and delta wolves in a pack of twelve wolves. Among the three dominant wolves, the alpha wolf is the leader of the group of wolves, a search agent to find the location of the prey, and gives a command to its subordinate wolves beta, delta, and the lowest level wolves called omega wolves. These omega wolves follow the orders obtained from three dominant wolves to encircle and attack the prey based on the location of the dominant wolves from the prey. Mathematically, GWO considers the locations of the dominant wolves as the best first three solutions. From these best three solutions, a new solution is found by changing the coefficient vectors A and C. In every iteration, the value of A is reduced from zero to two to ensure the position of the search agents goes near to prey for attacking and getting the global solution instead of the local solution. Similarly, the value of A>1 indicates, the search agent is moving away from the prey and search better prey, i.e., the best solution.

Many researchers proposed variations in grey wolf algorithms such as Enhanced Grey Wolf optimization (EGWO), Improved Grey wolf optimization (IGWO), and Modified Grey Wolf optimization (MGWO). These variations of GWO consider only a circle-shaped area for searching and attacking the prey in an unknown space. For example, in EGWO [7], the decay function values are in the range (0, 1) to give an equal number of operations for exploration and exploitations. In EGWO the search agent's new location is obtained from the alpha wolf's location instead of adding the locations of the dominant alpha, beta, and delta wolves as in GWO. The best solution is only from the alpha wolf because it knows the location of prey very well than the other two dominant wolves and finds the best solution. In IGWO [8], weights are added to dominant wolves, so the weight of the first dominant alpha wolf is more than the other two dominant wolves because the alpha wolf knows the location of prey very well than the other two dominant wolves and finds the best solution, here weights are multiplied with a position of each dominant wolf. In MGWO [9], the decay function is modified by adding an exponential function to allocate 70% of operations for exploration and 30% for exploitation, resulting in an excellent global minimum value.

The proposed Altered Grey wolf optimization method (AGWO) accepts the population from a two-dimensional array instead of from a linear array. The existing formula in GWO finds the best solution from a circle area is altered in AGWO to find the best solution in a closed space. In this proposed method, the exploration operations to find the location of the best solution cannot go beyond this enclosed area when searching for a global solution. The first three solutions namely alpha, beta, and delta are quickly found as the population of inputs is stored in a two-dimensional array format. Furthermore, the proposed method schedules both offloaded tasks from the mobile device and migrated tasks from the previous cloudlet in different VMS of the cloudlet. This method considers four parameters such as task execution time, the energy consumption of the mobile device, the Percentage of requests executed by the cloudlet, and the average scheduling delay of the cloudlet. The arrival time of the task and the task size is emphasized to schedule the big tasks and at the same time, the small tasks are also given equal importance. The small tasks that arrive at the earliest should not be starved for VM which is ensured by the fitness function of this proposed method. In this way, this method executes all size tasks, and big tasks are specially considered by scheduling them in Large VMs.

This paper is constructed as follows. Section II explains the related work of the various authors, section III illustrates the proposed new scheduling algorithm with a multi-objective function, its performance evaluation is depicted with charts by comparing with various algorithms in section IV and the conclusion with future work is described in section V.

II. Related Works

Several scheduling techniques have been proposed for scheduling the tasks in the cloudlet of MCC. For example, a new scheduling method is proposed [10] to save the cloudlet's power consumption using switches. These switches are acting as weights to send the task from the mobile device to the cloudlet. Furthermore, switches select the task using an optimized Convolution Neural Network (CNN) with an Improved BAT Algorithm. However, this method does not discuss the task allocation to individual virtual machines in the cloudlet. Enhanced Bee Colony Optimization [EBCO] algorithm for task scheduling is used [11] to reduce the energy of the mobile device. The task is represented by three parameters: task id, the execution time of the task, and the amount of data transmitted along with the task with precedence information. The assigned task size must be smaller than the cloudlet size, which represents the task's fitness value, and if the maximum size of the task is greater than the cloudlet size, then the task is migrated to the cloud. This algorithm addressed load balancing with constraints ready time and precedence relationship and does not consider dynamic scheduling of mobile tasks and VM scheduling within the cloudlet.

A Cooperative multi-task Ant Colony Optimization scheduling algorithm [12] is used to increase the profit. This scheduling problem is an optimization problem to reduce the total energy consumed by all the scheduled tasks, average load ratio, and Guarantee ratio. This algorithm selects the most suitable task based on the profit ratio; the service provider determines the cloudlets based on their available resources. However, this algorithm only considers profits, so some providers are idle without sufficient resources. In addition, some mobile devices do not offload tasks to the cloudlet due to unavailable resources.

A new framework is proposed [13] (EEMC) to find the energy required to execute a task in the cloud. This new framework is known as the Rule Generation-based Energy Estimation Model (RG-EEM). This framework first finds the execution time taken by one byte of the task to find the total execution time of the task. Then, a fixed threshold level is used to determine the energy consumption of the mobile device. The tasks whose energy consumption is below the threshold level are clustered and executed in the cloud using the Shortest Job First (SJF) algorithm. Nevertheless, this method often requires the latest task information to achieve the optimum value.

A task assignment method [14] is used to reduce task delay while allocating tasks to the small cell Base Stations (sBS), which work only indoors. When the user is visiting a mall, the map of the mall is sent to the user's mobile device before executing the task in sBSs. The centralized server analyzes the task; if it is small, it is allocated to the nearest sBS; otherwise, the task is split into multiple sub-tasks and executed in numerous sBSs. This method reduces task delay. Even though the tasks are allocated to the nearest sBS, the allocation decision is centralized; therefore, this method consumes time. This application is appropriate only for indoors, which always needs an internet connection and a map of the place. A task assignment algorithm [15] in the cloudlet with an Improved Differential Evolution optimization algorithm is used to schedule the tasks in the cloudlet. This algorithm modifies all the operators of the differential evolution algorithm such as subtraction, addition, and multiplication. The population vector used here is the k-bit vector in which the first-bit stores the information of the cloudlet assigned, the second-bit stores task-id, and other information is stored in each bit of the population vector. This algorithm schedules the tasks using the first-fit

strategy derived from the calculated cost function values of the tasks. This method reduces the total response time of all users and total network resource consumption. However, this algorithm does not consider mobile tasks and schedules for each VM.

A Genetic algorithm-based task scheduling method in MCC [16] reduces the cost and energy consumption of the mobile device with a response time as a deadline. If the latency and budget are greater than the deadline, the task is not offloaded; otherwise, the task is offloaded while considering its dependency and data transmission rate. However, this scheduling method does not prefer large tasks with big data. A dynamic task scheduling algorithm in mobile cloudlet [17] with the deep learning method introduced a new architecture. This method uses a controller in its architecture to offload the task either to the cloudlet, smartwatch, or mobile device. This architecture reduces the computation time of tasks and the energy of the mobile device. But, it does not consider tasks from the mobile device, the security of the tasks, and the server. A multi-objective task scheduling algorithm [18] is used for scheduling by combining Conditional Autoregressive Value at Risk (CAViaR) with Dragonfly Algorithm (DA). This algorithm allocates the tasks to the cloud that satisfy the fitness function. This algorithm increases resource utilization, and execution time and decreases the makespan, and energy of the mobile devices and the cloud server.

A practical algorithm [19] with improved particle swarm optimization (PSO) and simulated annealing (SA) algorithms is used to schedule the workflow tasks. It eliminated the local optimum in the PSO algorithm by introducing an insertion-based perturbation operator and swap-based perturbation operator in the SA algorithm. As a result, the optimum global value is obtained in both PSO and SA by a new method called Iterated Local Search (ILS) that reduces the makespan of workflow tasks. However, this algorithm does not consider the mobility of tasks and the task migration from one cloudlet to another. The tasks are assigned using an optimal task assignment method [20] with the Ant-Colony Artificial Bee Colony optimization algorithm. It minimizes the mobile device's average completion time, and power consumption and balances the load. This algorithm uses a queue decision generator to balance the load in the cloudlets, and the communicative time between the users and cloudlets is the fitness function. However, this method lacks a dynamic scheduling algorithm for mobile tasks. A framework [21] is proposed to allocate the task based on the user's budget constraints to increase revenue; the service provider employed a one-round approximation algorithm to schedule tasks in the cloudlet. However, this framework does not consider the quality of services.

A. Motivation and Challenges

The extensive literature review of different authors discussed above has shown some challenges in MCC that are as follows:

1. The scheduling algorithms did not allocate the task to each virtual machine within the cloudlet. The discussed articles in the literature review consider the task allocation in the cloudlet only.
2. In the scheduling algorithms, the decision to allocate the task to multiple cloudlets is taken by one server and not by each cloudlet.
3. The scheduling algorithms did not give importance to the big tasks and the mobility of tasks.

This article focuses on the challenges mentioned above with a dynamic scheduling technique that gives importance to big tasks from mobile devices. It allocates these tasks to VMS based on the processing speed of the VMS within a cloudlet. The decision to allocate the task to the VM is taken within the cloudlet and not by other cloudlets. So, the execution time of the task is reduced. This article reduces the execution time of the offloaded task from the mobile device and the power consumption on that mobile device.

III. PROPOSED METHOD

MCC offloads the tasks to the cloudlet to reduce the execution time of the offloaded task and the energy used by the mobile device. Initially, the mobile device connects to a nearby cloudlet and as the mobile device moves, it is disconnected from the existing cloudlet and connected to the next nearby cloudlet. As a result, a partially executed offloaded task is migrated to the new cloudlet. In the new cloudlet, this migrated task is scheduled using this proposed method to enhance the user's QoE (Quality of Experience).

A. Grey Wolf Optimizer

The proposed method uses the Altered Grey Wolf optimization algorithm [AGWO] to schedule the tasks in the VMS of the cloudlet that accept inputs from the two-dimensional array. In GWO, the dominant alpha, beta, and delta wolves find the location of the prey d, which is the best solution based on the distance between each of the three wolves and the prey using equation 1. Xp is the position of prey. X is the position of the grey wolves and t is the iteration. A and C are coefficient vectors as in GWO. The new prey is discovered by calculating the distance between the grey wolves, and the location of the prey is determined using equation 2. In the next iteration $t+1$, the position of the alpha, beta, and grey wolves are updated using equation 2. The coefficient vector A decides whether grey wolves attack or diverge the prey. The coefficient vector C adds random weights to the prey or the solution. It takes weights between 0 and 2 to determine the position of the prey reached easily or with difficulty. If $C < 1$, the prey is easily reached there is no obstacle in reaching the prey, which is used in equation 1. If $C > 1$, there are obstacles in reaching the prey.

$$d = C.Xp\,(t) - X(t) \quad (1)$$

$$X(t+1) = Xp\,(t) - A.d \quad (2)$$

B. Altered Grey Wolf Optimization Algorithm

This article proposed a new scheduling algorithm with an altered Grey wolf optimization algorithm that accepts the tasks from the mobile device to be scheduled. The tasks are stored in a two-dimensional array format. The GWO is altered to accept the tasks in a two-dimensional array and schedule these tasks

using the Altered Grey wolf Optimization algorithm (AGWO). The scheduling algorithm (AGWO), a meta-heuristic algorithm, uses a multi-objective function in equation 3 to minimize the task execution time of n tasks and the mobile device's power consumption.

Minimize

$$Z = \sum_{t=1}^{n} xt + pw \quad (3)$$

Equation 1 is used to find the location between dominant wolves, and the prey is altered in this method from the original equation because the searching region is a two-dimensional array and not an open circle like GWO. A task's arrival time is represented by a thousand rows numbered from one millisecond to one thousand milliseconds. The two-dimensional array format is given in table 1 and named as task_2d (1000, 6). The column of the two-dimensional array indicates the velocity of the mobile device; the first column contains the migrated job from the previous cloudlet, the second column indicates the tasks with a velocity of 1-2 milliseconds, the third column indicates 3-4 milliseconds, the fourth column indicates 5-6 milliseconds, the fifth column indicates 7-8 milliseconds and the sixth column indicates 9-10 milliseconds, respectively. The last column stores the task of stationary devices and a velocity greater than 10ms. So, the two-dimensional array consists of 1000 rows and seven columns. The pseudo-code of the algorithm is given in Algorithm 1.

TABLE I
TWO-DIMENSIONAL ARRAY TASK_2D (1000,6)

The speed of the mobile device (m/s)	Arrival time of the task in millisec (ms)						
	migrated task	1-2 ms	2-4 ms	5-6 ms	7-8 ms	9-10 ms	>10 ms
	Task size in kilobytes (KB)						
1	8	12	13	4	100	120	200
2	10	14	15	4	400	220	400
3	800	450	50	400	100	120	200
999	100	4	5	4	6	7	85
1000	300	45	7	8	12	17	53

This algorithm saves the tasks in the two-dimensional array's corresponding column's velocity and row's arrival time. First, the virtual machines in the cloudlet are divided into seven groups based on their CPU size: the first group has a size of 500 MIPS, the second group has a size of 1000 MIPS, the third group has a size of 1500 MIPS, the fourth group has a size of 2000 MIPS, the fifth group has a size of 2500 MIPS, the sixth group has a size of 3000 MIPS, and the seventh group has a size greater than 3500 MIPS. Next, find the largest VMS group with more virtual machines from the first six VMS groups, excluding the last VMS group, and divide this largest group of VMS into two groups. Therefore, we have eight VMS groups, including the last VMS group, and arrange the groups of VMs in ascending order, and in each group, get the number of VMs. Then, starting with the first row, find the smallest tasks (less than 8192 bytes), choose them in FCFS (First Come, First Serve) order, and distribute them according to the following technique. The first column of tasks corresponds to the first group of VMS, the second column to the second group of VMS, the third column to the third group of VMS, the fourth column to the fourth group of VMS, and the fifth column to the fifth group of VMS, the sixth column to the sixth group of VMS, and the seventh column to the seventh group of VMS, except for the last group of VMS. The largest tasks satisfying the fitness equation 4 are discovered using the arrival time and task size from their corresponding row and column. The alpha, beta, and delta wolves are the first three minimum fitness values of the largest jobs discovered, from which the new best task is chosen using the AGWO algorithm.

$$f = min\, f(x) = \sum_{j=1}^{n} at/ts \quad (4)$$

C. Encircling the Prey

The parameters A and C are used to determine whether to attack or leave the prey, and A is used to encircle the prey, which is the current iteration's selected task, and the task is assigned if $A < 1$ and abandoned by grey wolves if $A > 1$.

$$\begin{aligned} alpha\_dist_x &= (C1 + (X_{alphax} - currentX_x))\%rs, \\ alpha\_dist_y &= (C1 + (X_{alphay} - currentX_y))\%cs, \\ beta\_dist_x &= (C2 + (X_{betax} - currentX_x))\%rs, \\ beta\_dist_y &= (C2 + (X_{betay} - currentX_y))\%cs, \\ delta\_dist_x &= (C3 + (X_{deltax} - currentX_x))\%rs, \\ delta\_dist_y &= (C3 + (X_{deltay} - currentX_y)\%cs \end{aligned} \quad (5)$$

The distance between the search agents alpha, beta, delta wolves, and the prey is calculated using equation 5. To ensure the x and y coordinates of the computed distance should not cross the boundary of two-dimensional array locations are confirmed in equation 5 by dividing the x-coordinate by row size and y-coordinate by column size for all three wolves.

$$\begin{aligned} x1_x &= X_{alphax} - (A1 \times alpha\_dist_x), \\ x1_y &= X_{alphay} - (A1 \times alpha\_dist_y), \\ x2_x &= X_{betax} - (A2 \times beta\_dist_x), \\ x2_y &= X_{betay} - (A2 \times beta\_dist_y), \\ x3_x &= X_{deltax} - (A3 \times delta\_dist_x), \\ x3_y &= X_{deltay} - \left(A3 \times delta_{dist_y}\right). \end{aligned} \quad (6)$$

Equation 6 gives the first three solutions. The first solution $x1_x$ and $x1_y$ represent the x and y coordinates of the first solution. It is obtained by finding the distance between the alpha wolf's position and the distance between the prey and the alpha wolf multiplied by vector A. In the same way, the second solution $x2$ is obtained by finding the distance between the beta wolf position and the distance between prey and the beta wolf multiplied by vector A. The third solution $x3$ is also obtained by finding the distance between the delta wolf position and the distance between prey and the delta wolf multiplied by vector A. It contains the value of vector $A < 1$, decides to attack the prey and is used to decide the solution or prey is the best prey or solution.

$$A = 2a \times r1 - a \quad (7)$$

$$C = 2 \times r2 \quad (8)$$

Equation 7 yields a value of A, with component a decreasing from two to zero in each iteration; if the value of $A > 1$, the grey wolves move away or diverge from the prey and ensures that this prey or solution is not the best one. Random vectors $r1$ and $r2$ take values between 0 and 1. Finally, using equation 8, vector C lends weight to the prey to assure the global optimum in every iteration, and the new solution is obtained by taking the average of $x1$, $x2$ and $x3$ distances in equation 9. The next solution is obtained by finding the average of the first three solutions. Because the first three solutions represent the knowledge of dominant wolves such as alpha, beta, and delta wolves. This knowledge is the solution obtained from these wolves. These three solutions will have the clue for the next solution. So, the next solution is obtained from the average value of the first 3 solutions. $Xnew_x$ is the x coordinate of the new solution is obtained by taking the average value of x coordinates of alpha, beta, and delta solutions. $Xnew_y$ is the y coordinate of the new solution is obtained by taking the average value of the y coordinates of alpha, beta, and delta solutions

$$Xnew_x = (X1_x + X2_x + X3_x)/3,$$

$$Xnew_y = (X1_y + X2_y + X3_y)/3 \quad (9)$$

$$T_{ofvn\_2} = \sum_{k=1}^{n-1} T_{ofvm\_k} + \sum_{k=1}^{n-1} Tot_{exk} \quad (10)$$

$$T_{pwvn\_2} = \sum_{k=1}^{n-1} T_{pvm\_k} + \sum_{k=1}^{n-1} Tot_{pvmexk} \quad (11)$$

ALGORITHM I
NOTATION TASK SCHEDULING WITH ALTERED GREY WOLF OPTIMIZATION FOR TWO-DIMENSIONAL ARRAY INPUTS

TSAGWO (Input: Task_2d (A, B);
Input: Tasks stored in two-dimensional array Task_2d (A, B) with Their arrival time in A and their velocity in B.
Output: The best task from Task_2d (A, B)

1. AGWO population ← Task_2d (A, B) where A = 1, 2,…,1000 & B=1,2,.,7
2. Divide the vms of cloudlet into vms-groups G [7].
   {
   G1 ←vms_size (500 MIPS)
   G2 ←vms_size (1000 MIPS)
   G3 ←vms_size (1500 MIPS)
   G4 ←vms_size (2000 MIPS)
   G5 ←vms_size (2500 MIPS)
   G6 ←vms_size (3000 MIPS)
   G7 ←vms_size > (3500 MIPS)
   }
3. while (vms-groups<=6) /* Equal number of vms */
   {
   If (any vms-groups have more vms) then
   Divide vms-groups into two groups
   }
4. Sort(G[7],G[7]) /* Sort the vms-groups in ascending order */
5. While (true) {
6. Initialize a, A, and C, Task_2d (A, B)
7. Declare $X\alpha$, $X\beta$, $X\delta$ are the best search agents
8. Find the task of size <8192 bytes from each row
9. G[1..6] ←Allocate the tasks of size<8192 bytes from Task_2d [A,B] in FCFS
10. CALCUL_OBJFUN(G[1..6]), Z) /* Call Procedure */
11. Let $X\alpha$, $X\beta$, $X\delta$ be the first, second, and third minimum fitness value
12. Let t1,t2,t3 be the first 3 big tasks of size > 65536 bytes
13. $X\alpha$ ← Task_2d [A](t1) / Task_2d [B] (t1)
14. $X\beta$ ← Task_2d [A](t2) / Task_2d [B] (t2)
15. $X\delta$ ← Task_2d [A](t3) / Task_2d [B] (t3)
16. G[7] ← $X\alpha$, $X\beta$, $X\delta$
17. while (m < maximum iterations)
18. {
19. for each search agent
20. {
21. UPDATE_DIST($X\alpha$, $X\beta$, $X\delta$, ($X1_x$, $X1_y$),($X2_x$, $X2_y$), ($X3_x$, $X3_y$)) /* Call Procedure */
22. $Xnew_x = (X1_x + X2_x + X3_x) / 3$, $Xnew_y = (X1_y + X2_y + X3_y) / 3$
23. }
24. A1,A2,A3 ← 2a.r1.a
25. C1,C2,C3 ← 2.r2
26. if (($Xnew_x$ $Xnew_y$)< $X\alpha$)
27. $X\alpha$ ← ($Xnew_x$ $Xnew_y$) /*Allocate new solution*/
28. m=m+1
29. }
28. return $X\alpha$ /*new task for allocation */
30. }

PROCEDURE CALCUL_OBJFUN (Input: G [1...6]), Output: Z)

/* calculate the execution time of the allotted task using equation 10 and the power Consumption of the mobile device using equation 11 */

a. n ← the number of cloudlets traversed
b. ofvn ← Total execution time of task
c. pwvn ← Total power consumption of the mobile device
d. pvm ← Power consumption of the mobile device during connection
e. Pvmex ← Power consumption of the mobile device during execution
f. For (k=1,K<n, K++)
g. {
h. ofvn= offloading time of task in k+ the execution time of task in k
i. Pow= Pvm in k + Pvmex in k
j. }.

PROCEDURE UPDATE_DIST ($X\alpha$, $X\beta$, $X\delta$, ($X1_x$, $X1_y$),($X2_x$, $X2_y$), ($X3_x$, $X3_y$))
/* update alpha-dist, beta-dist, delta-dist and X1, X2, X3 by equ 5and 6 */
a. A ← row size of the array Task_2d (A, B))
b. B ← column size of the array Task_2d (A, B))
c. αx = alpha-dist of x, αy = alpha-dist of y
d. βx = beta-dist of x, βy = beta-dist of y
e. δx = delta-dist of x, δy = delta-dist of y
f. αx ← (x coordinate of $X\alpha$ – x coordinate of prey) /A
g. αy ← (y coordinate of $X\alpha$ – y coordinate of prey) / B
h. βx ← (x coordinate of $X\beta$ – x coordinate of prey) /A
i. βy ← (y coordinate of $X\beta$ – y coordinate of prey) / B
j. δx← (x coordinate of $X\delta$ – x coordinate of prey) /A
k. δy ← (y coordinate of $X\delta$ – y coordinate of prey) / B
l. $X1_x$ = $X\alpha$-A* alpha-dist of x, $X1_y$= $X\alpha$-A* alpha-dist of y
m. $X2_x$ = $X\beta$ -A* beta -dist of x, $X2_y$= $X\beta$ -A* beta -dist of y
n. $X3_x$ = $X\delta$ -A* delta -dist of x, $X3_y$= $X\delta$ -A* delta -dist of y

In each iteration, the new task's fitness value is compared against the current task, and the task with the lowest fitness value wins. The best task is assigned to the cloudlet's last group of VMs after the maximum iteration. The execution time and power consumption of the allotted tasks are found using equations 10 and 11 from our previous research article Mobility and Execution Time Aware Task Offloading method (METATO) [22]. This way, the tasks within one second are assigned to the cloudlet and continue with the next second of tasks. The notations and explanations are given in Table II.

IV. PERFORMANCE EVALUATION

A computer Powered by Intel Core i7 3770K 3.5GHz Quad-Core processor with 16GB RAM and 1TB storage is used to develop and test the proposed method. In addition, the visual C++ programming language is used to code the functional modules of the proposed method and sent to the cluster manager of the Apache Spark framework with default configurations in cluster mode from a Common Gateway Interface (CGI). A dedicated User Interface (UI) is designed as a CGI using Visual Studio Integrated Development Environment. The UI is designed similarly to a Desktop Client for a Cloud Service. The

UI is responsible for loading the programs such as AGWO, GWO, IGWO, MGWO, EGWO, EEMC, OTAQAA, etc. through CGI to the Apache Spark Framework and returning the result to the user. To execute big tasks from mobile devices this Apache Spark Framework is a suitable tool and is used as a cloudlet. The CGI executes and evaluates the performance of all the programs and sends the comparison results of all the programs as output in charts. The Apache Spark Framework receives the mobile device data such as mobility, task, and size from the open data mobile dataset. In the Apache Spark framework, the cluster manager is the scheduler and allocates the task to the worker nodes according to the program. The experimental setup is given in Figure 1.

The analysis is carried out in two sets of comparisons. In the first set, the performance of the proposed AGWO and variants techniques GWO, MGWO, IGWO, and EGWO are analyzed. In addition, the parameters such as the Average Request Execution time of mobile nodes, Average power consumption of the mobile node, and Percentage of requests executed are analyzed in this article. Another set of comparisons includes contributions from other researchers.

TABLE II
LIST OF NOTATIONS

Notation	Description
Xt	Task execution time
Pw	Mobile device energy consumption.
T	The variable represents the applications
N	Number of applications
alpha-$dist_x$,,alpha-$dist_y$	Distance between the alpha wolf and prey
A1, A2, A3, C1, c2, c3	Coefficient vectors
beta-$dist_x$, beta-$dist_y$	Distance between the beta wolf and prey
delta-$dist_x$, delta-$dist_y$	Distance between delta wolf and prey
x-$alpha_x$, x-$alpha_y$	Position of alpha wolf
current-x_x, current-x_y	Position of the current wolf in each iteration
X1 + X2 + X3	Position of best three search agents
Task_2d (A,B)	A two-dimensional array with row size A and Column size B
G [7]	Vms of a cloudlet are divided into 7 groups
Tot_{ex}	The total execution time of the VM migrated task
T_{ofvn-2}	The offloading time of the VM migrated task
T_{pvm-k}	Power consumption required by the mobile device to connect with a cloudlet-k
Tot_{pvmexk}	Power consumption required by the mobile device to execute the task in a cloudlet-k
T_{pwvn-2}	Power required by the mobile device to offload and execute the task in all the cloudlets
Rs	Row size
Cs	Column size

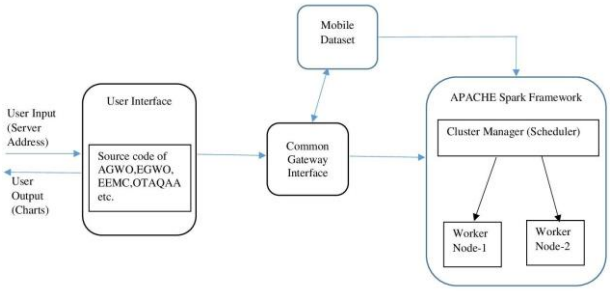
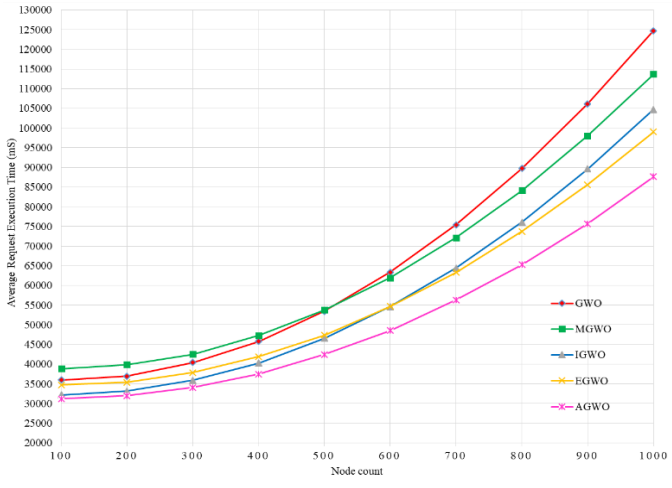


Fig.1. Experiment Setup using Apache Spark Framework as Cloudlet

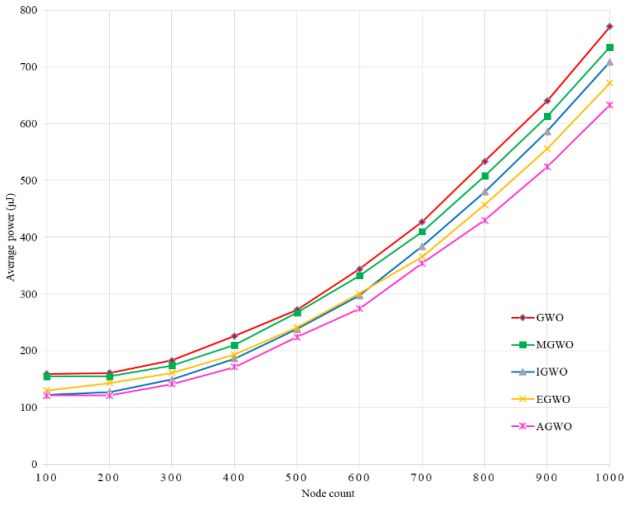
Figure 2a shows that the average request execution time of the mobile nodes is found by varying numbers of nodes. When the number of nodes exceeds 600, the average request execution time is reduced gradually using this proposed method, AGWO. This proposed method is suitable for scheduling more tasks efficiently and avoids the idle time of VMs in the cloudlet.

The mobile device's average power consumption is depicted in Figure 2b. When the number of nodes is small, the power consumption of the mobile nodes is nearly identical for all optimization algorithms, but when the number of nodes grows, the suggested method AGWO reduces the mobile node's power consumption more than other optimization methods (GWO, MGWO, IGWO, and EGWO). When the number of nodes exceeds 800, additional tasks are scheduled, and this approach allocates tasks larger than 65535 bytes in a VMS with a large capacity of 3000 MIPS and RAM greater than 2GB. When allocating large-size jobs, the mobile nodes consume little power using this proposed method.

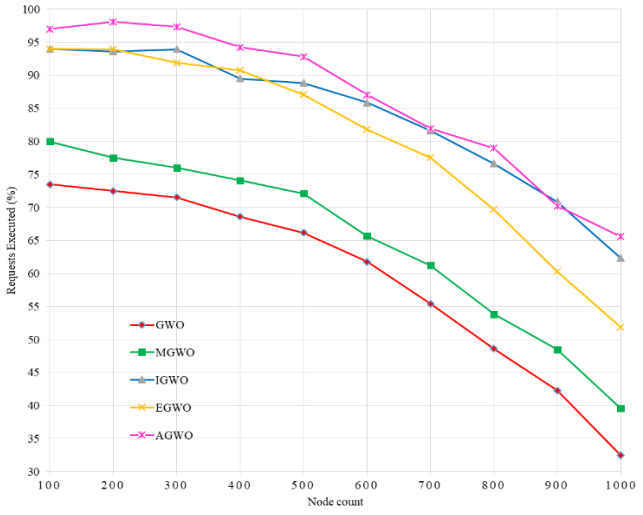
Figure 2c illustrates that a high percentage of requests from the mobile device are executed in the proposed method than in other optimization algorithms (GWO, MGWO, IGWO, and EGWO). Because AGWO searches in known locations, when the number of nodes exceeds 900, the proposed methods perform better than IGWO. Figure 2d illustrates that for a



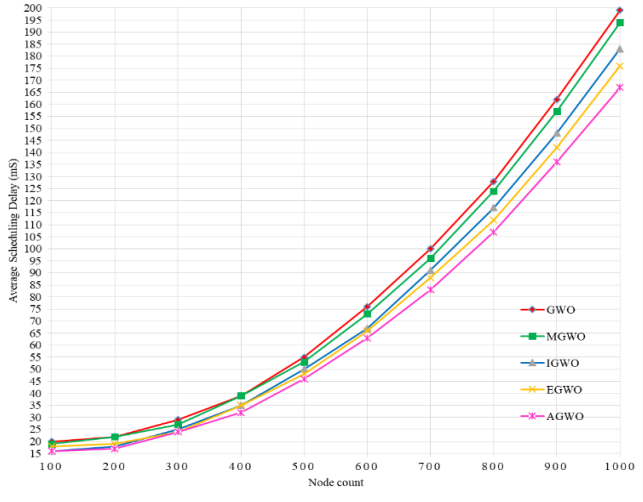
(a) Average request execution time of mobile nodes



(b) Average power consumption of mobile nodes



(c) Percentage of requests executed by the cloudlet



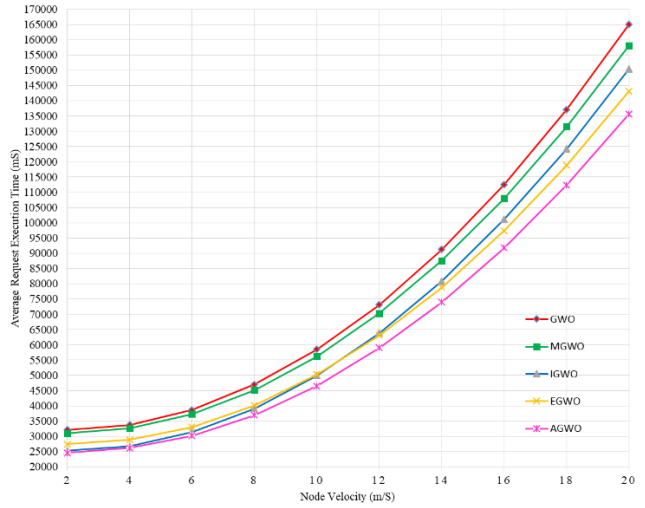
(d) Average scheduling delay by the cloudlet

Fig. 2. Effect of different numbers of mobile nodes (100 to 1000)

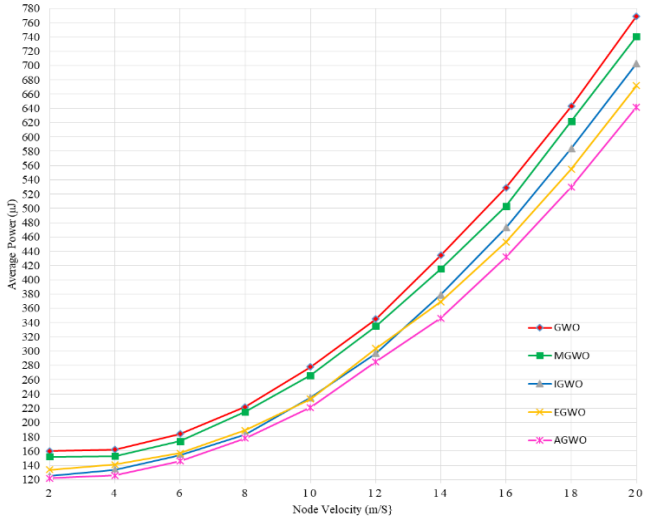
smaller number of nodes, the average scheduling delay time is nearly comparable for all optimization strategies. When the number of nodes grows, however, the suggested method requires less average scheduling delay time than existing optimization algorithms and a larger number of jobs are efficiently distributed to the appropriate VMS within the cloudlet based on their size.

Figure 3a shows that the average execution time of the mobile nodes has decreased, even though the velocity of mobile devices has increased by up to 20ms. The migration of tasks rises as the speed of the mobile device increases. Because task sizes smaller than 8192 bytes are arranged in a first-come, first-served order to their corresponding appropriate VMS, the average request execution time is lowered.

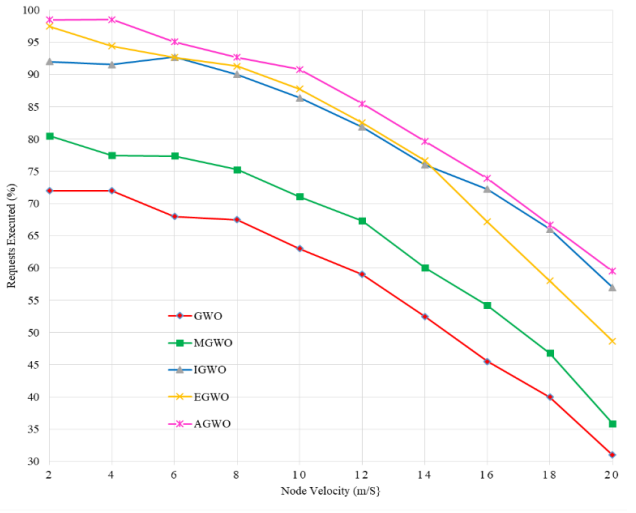
Figure 3b shows that when the devices' velocity increases, the mobile device's average power consumption reduces, and the proposed method performs better for high-velocity mobile devices. According to the algorithm, the tasks of mobile device velocity greater than 10ms are allocated to the last group of VMS, which is a group of large-size VMS. So, the tasks finish



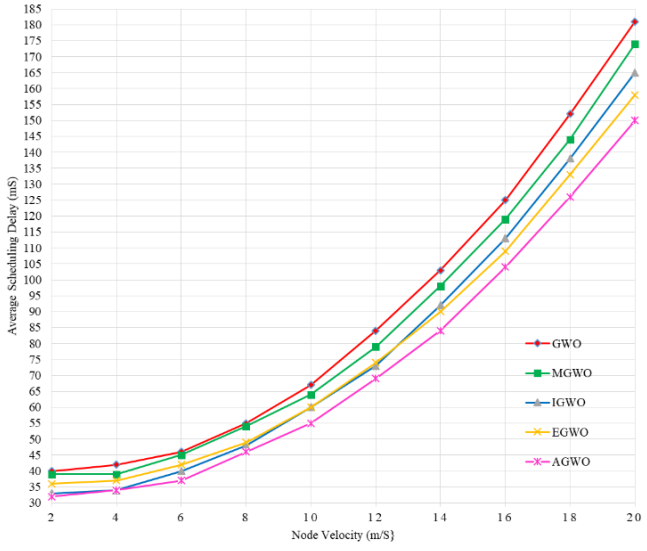
(a) Average request execution time of the mobile nodes



(b) Average power consumption of the mobile node



(c) Percentage of requests executed by the cloudlet



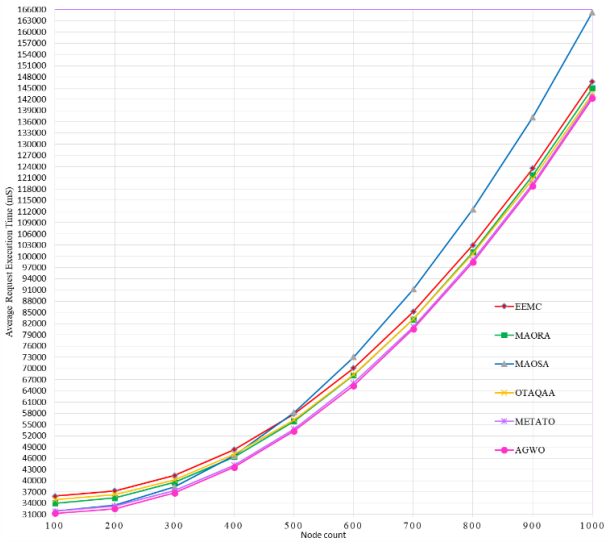
(d) Average scheduling delay of the cloudlet

Fig. 3. Effect of different mobility speeds (random speed)

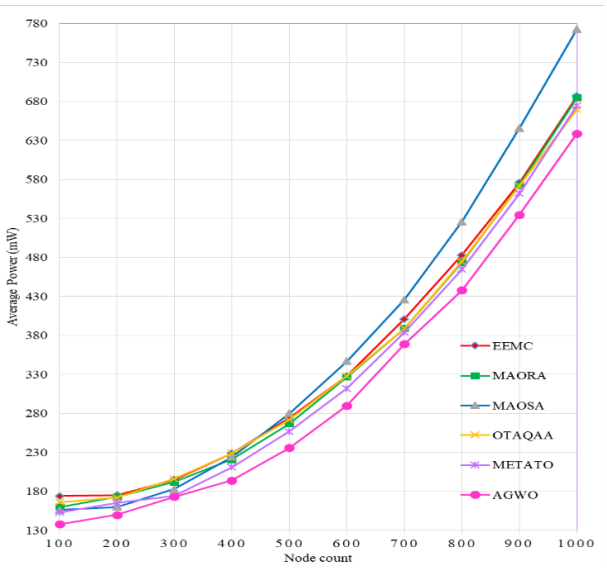
their execution within the allotted cloudlet without VM migration resulting in reduced execution time. As a result, AGWO processes more requests whether velocity increases or decreases. The cloudlet's average scheduling delay is also lowered compared to existing optimization algorithms, as in figures 3c and 3d. Furthermore, this proposed method allocates various-size tasks to appropriate VMS based on their velocity as high-velocity tasks are allocated to large-size VMS.

The performance of the proposed AGWO and existing techniques EEMC, METATO, MAORA [23], MAOSA [24], and OTAQAA are analyzed in this section. The same parameters are also used to show that AGWO performs better in the second set of comparisons. Figure. 4a. Shows that scheduling more tasks efficiently is possible using this proposed method, which eliminates the idle time of VMs in the cloudlet. The average power consumption of a mobile device is illustrated in figure 4b.

A high percentage of requests from the mobile device are executed in the proposed method than in other existing methods, as shown in figure 4c and the average scheduling delay time is also reduced as in figure 4d. During higher velocities, also AGWO performs best, as depicted in figures 5a, 5b, and 5c. As in Figure 5a, when the speed increases, the task execution time is reduced because big tasks and small tasks are treated equally along with their arrival times and less migration rate.

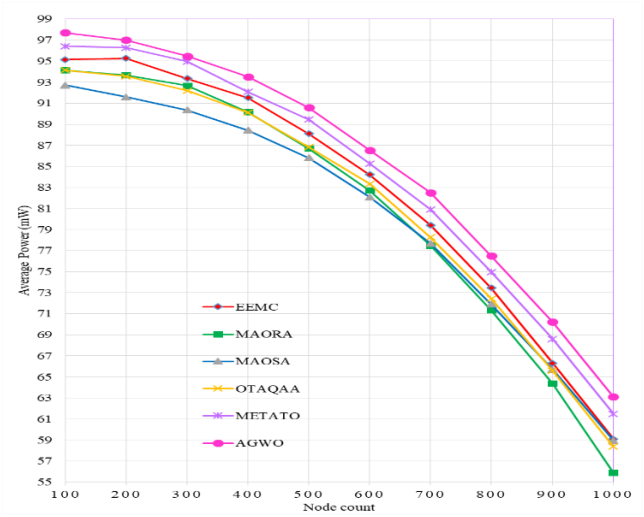


(a) Average request execution time of mobile nodes

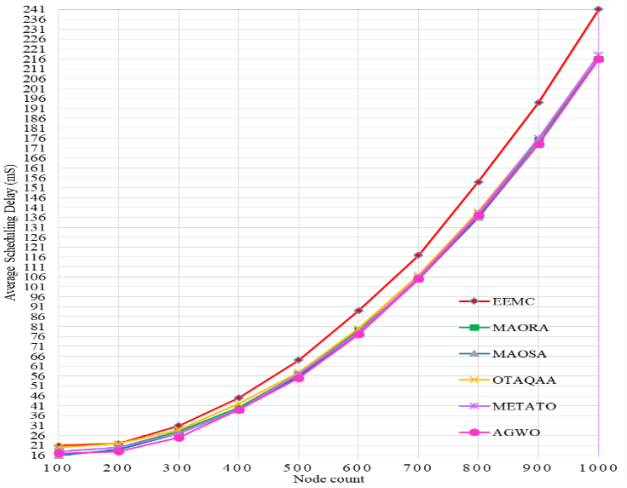


(b) Average power consumption of mobile nodes

Whenever the number of nodes is small, the power consumption of the mobile nodes is nearly identical to those of all existing methods. Allocating large jobs with this method requires relatively little power consumption from the mobile nodes. For a smaller number of nodes. Figure 5d shows that the average scheduling delay time is less when the speed of the device is less. In any case, the suggested method requires less average delay time than existing techniques with a growing number of nodes. By determining the task size, more tasks can be distributed efficiently to the appropriate VMS within the cloudlet based on their size.

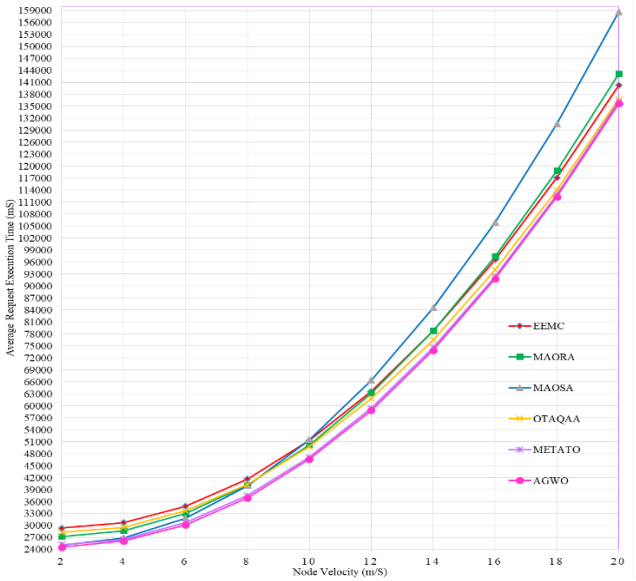


(c) Percentage of requests executed by the cloudlet

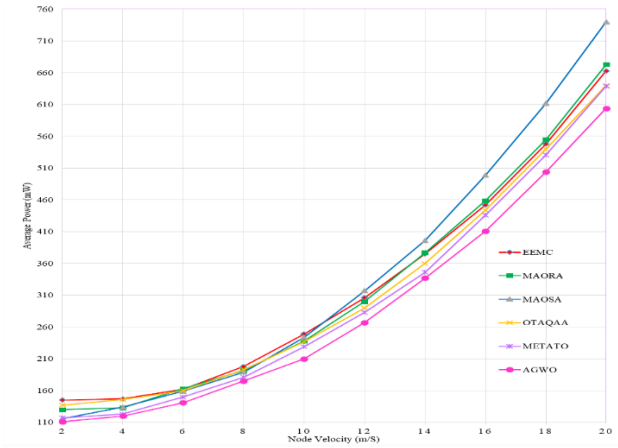


(d) Average scheduling delay by the cloudlet

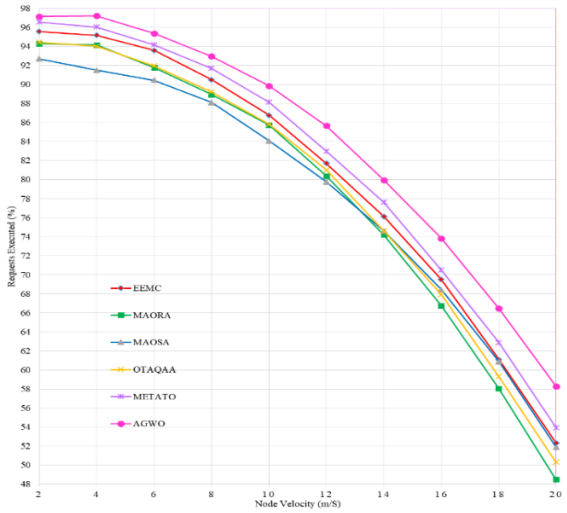
Fig. 4. Effect of different numbers of mobile nodes (10 to 100)



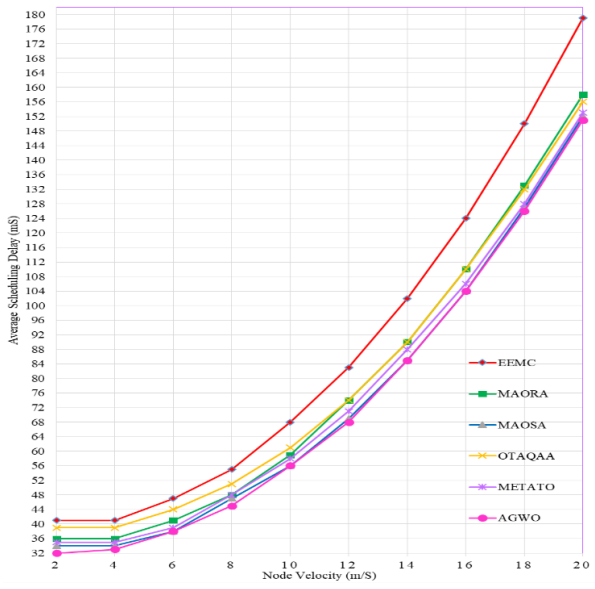
(a) Average request execution time of mobile nodes



(b) Average power consumption of mobile nodes



(e) Percentage of requests executed by the cloudlet



(f) Average scheduling delay by the cloudlet

Fig. 5. Effect of different mobility speeds (random speed)

V. Conclusion

Task scheduling with altered Grey wolf optimization for two-dimensional array inputs (TSAGWOMCC) is proposed to allocate tasks in different sizes, including big tasks, and is an NP-hard problem. The experiment results show that the proposed method reduces the average power consumption of the mobile device, the average scheduling delay of tasks in the cloudlet, and the average execution time of the task. The percentage of requests executed by the cloudlet is also increased through this method. The experiment is conducted by varying numbers of nodes and with the different velocities of mobile nodes. Large tasks greater than 65536 bytes are specially treated by allocating them to VMS of enormous size. So the migration rate is reduced, which in turn reduces the power consumption of the mobile device and increases the Percentage of requests executed by the cloudlet. Parallel execution of the sub-tasks of the same task in the heterogeneous cloudlet will be the future work. Balancing the loads in different cloudlets and reducing the dropout rate of tasks will also be considered in the next work.

References

[1] M. Curiel and L. Flórez-Valencia, "Challenges in Processing Medical Images in Mobile Devices, Trends and Advancements of Image Processing and Its Applications", EAI / Springer Innovations in Communication and Computing, pp-31-51.2022., https://doi.org/10.1007/978-3-030-75945-2_2

[2] S. Singh and I. Chana, "Q-aware: quality of service-based cloud resource provisioning, Computers & Electrical Engineering", vol.47, pp. 138-160, 2015. https://doi.org/10.1016/j.compeleceng.2015.02.003

[3] C. Tang, M. Hao, X. Wei and W. Chen, "Energy-aware task scheduling in mobile cloud computing", Distributed and Parallel Databases, vol.36, pp.529-533, 2018. https://doi.org/10.1007/s10619-018-7231-7

[4] S. Mirjalili, S. M. Mirjalili and A. Lewis, "Grey Wolf Optimizer", Advances in Engineering Software, vol.69, pp.46-61, 2014. https://doi.org/10.1016/j.advengsoft.2013.12.007

[5] S. Li, H. Chen, M. Wang, A. Asghar Heidari and S. Mirjalili, "Slime mould algorithm: A new method for stochastic optimization", Future Generation Computer Systems, vol.111, Oct., pp. 300-323, 2020. https://doi.org/10.1016/j.future.2020.03.055.

[6] D. Fouskakis and D. Draper, "Stochastic Optimization: A Review", International Statistical Review, vol.70, no.3. Dec., pp. 315-349, 2002. https://doi.org/10.2307/1403861

[7] H. Joshi and S. Arora, "Enhanced Grey Wolf Optimization Algorithm for Global Optimization", Fundamenta Informaticae, vol.153, no.3, Jun., pp.235-264, 2017. https://doi.org/10.3233/FI-2017-1539

[8] Z.- M. Gao and J. Zhao, "An Improved Grey Wolf Optimization Algorithm with Variable Weights", Computational Intelligence and Neuro science, vol.2019, Jun., 2019. https : //doi.org /10.1155 /2019 /2981282

[9] N. Mittal, U. Singh, and B. Singh Sohi, "Modified Grey Wolf Optimizer for Global Engineering Optimization", Applied Computational Intelligence and Soft Computing, vol. 2016, 2016., https://doi.org/10.1155/2016/7950348

[10] P. Akki and V. Vijayarajan, "Energy Efficient Resource Scheduling Using Optimization Based Neural Network in Mobile Cloud Computing", Wireless Personal Communications, vol.114, no.2,pp. 1785-1804,2020. https://link.springer.com/article/10.1007/s11277-020-07448-2

[11] C. Arun and K. Prabu, "A multi-objective EBCO-TS algorithm for efficient task scheduling in mobile cloud computing", International Journal of Networking and Virtual Organizations, Vol.22, no.4, pp.366 – 386,2020. https://dx.doi.org/10.1504/IJNVO.2020.107570

[12] T. Wang, X. Wei, C. Tang and J. Fan, "Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints", Peer-to-Peer Networking and Applications, vol.11, no.4, pp. 793-807,2018. https://link.springer.com/article/10.1007/s12083-017-0561-9

[13] S. E. Veerappa Dinesh, and K. Valarmathi, "A novel energy estimation model for constraint based task offloading in mobile cloud computing", Journal of Ambient Intelligence and Humanized Computing, Vol.11, pp.5477–5486, 2020. https://doi.org/10.1007/s12652-020-01903-5

[14] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni and R. Wang, "User mobility aware task assignment for Mobile Edge Computing", Future Generation Computer Systems, Vol.85, no.c, Aug., pp.1-8, 2018., https://doi.org/10.1016/j.future.2018.02.014

[15] A. Zhou, Y. Li and J. Li, "Efficient request assignment algorithm in mobile cloud computing environment", International Journal of Web and Grid Services, Vol.14, no.4, pp. 335-351, 2018. https: //dx.doi.org/10.1504/IJWGS.2018.095656

[16] C. Tang, S. Xiao, X. Wei, M. Hao and W. Chen, "Energy-efficient and Deadline-satisfied Task Scheduling in Mobile Cloud Computing", In proc. IEEE International Conference on Big Data and Smart Computing '01, 2018. https: //doi.org /10.1109/ BigComp .2018.00037

[17] D. S. Rani, M. Pounambal, "Deep learning based dynamic task offoading in mobile cloudlet environments", Evolutionary Intelligence, Vol.14, pp.499-507, 2019. https://dx.doi.org/10.1007/s12065-019-00284-9

[18] M. Garg and R. Nath, "Auto regressive Dragonfly Optimization for Multiobjective Task Scheduling (ADO-MTS) in Mobile Cloud Computing", Journal of Engineering Research, Vol. 8, no. 3, pp. 71-90, 2020. https://doi.org/10.36909/jer.v8i3.7643

[19] H. Li, Y. Zhu, M. Zhou and Y. Dong, "Effective Algorithms for Scheduling Workflow Tasks on Mobile Clouds", Journal of Circuits, Systems, and Computers, vol.29, no.16, 2020. https: //doi.org/10.1142/S0218126620502552

[20] V. Sundararaj,"Optimal Task Assignment in Mobile Cloud Computing by Queue Based Ant-Bee Algorithm", Wireless Personal Communications: An International Journal , vol.104, no.1, Jan., pp. 173–197, 2019. https://doi.org/10.1007/s11277-018-6014-9

[21] Y. He, L. Ma, R. Zhou, C. Huang, and Z. Li, "Online Task Allocation in Mobile Cloud Computing with Budget Constraints", Computer Networks, vol.151, no.14, Mar., pp.42-51, 2019., https://doi.org/10.1016/j.comnet.2019.01.003

[22] J. A. Mary, and A. Aloysius, "Mobility and Execution time aware task offloading in Mobile Cloud Computing", International Journal of Interactive Mobile Technologies, Vol.16, no.15, pp.30-45, 2022. https://doi.org/10.3991/ijim.v16i15.31589

[23] A. Enayet, Md. A. Razzaque, M. M. Hassan, A. Alamri, and G. Fortino, "A Mobility-Aware Optimal Resource Allocation Architecture for Big Data Task Execution on Mobile Cloud in Smart Cities", IEEE Communications Magazine ,Vol. 56, no.2, Feb., pp.110-117, 2018. https://doi.org/10.1109/MCOM.2018.1700293

[24] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "MuSIC: Mobility-Aware Optimal Service Allocation in Mobile Cloud Computing", In Proc. IEEE Sixth International Conference on Cloud Computing '07, 2013. https://doi.org/10.1109/CLOUD.2013.100.

J. Arockia Mary is doing her Doctoral Degree in Computer Science at St. Joseph's College (Affiliated to Bharathidasan University), Tiruchirappalli, Tamil Nadu, India. She has more than 12 years of teaching and research experience. She is currently working as an Assistant Professor, at the Department of Computer Applications, Holy Cross College, Tiruchirappalli, Tamil Nadu, India. She has published papers in international journals. She has presented papers at international conferences. Presently she is doing her research on Mobile Cloud computing. (email:jarockia79@gmail.com)

A. Aloysius is working as an Assistant Professor in the Department of Computer Science, St. Joseph's College (Affiliated to Bharathidasan University), Tiruchirappalli, Tamil Nadu, India. He has 22 years of experience in teaching and research. He has published many research articles in national and international conferences and journals. He also presented research articles at International Conferences on Computational Intelligence and Cognitive Informatics in Indonesia. He has acted as a chairperson for many national and international conferences. His current research area includes cognitive aspects in software design, big data, sentiment analysis and cloud computing. (email:aloysius1972@gmail.com)