Journal of Engineering and Applied Sciences 14 (8): 2716-2728, 2019
ISSN: 1816-949X

Task Scheduling for Mobile Cloud Computing Using Multi-Objective EBCO-TS Algorithm

C. Arun and K. Prabu
Department of Computer Science, Sudharsan College of Arts and Science,
Pudukkottai, Tamil Nadu, India

Abstract: Based on certain defects encountered in mobile devices, like insufficient storage space, limited battery energy, mobile applications faces numerous confronts in energy management, mobility management, security issues and so on. This leads to the emergence of the new computing paradigm known as Mobile Cloud Computing (MCC). This kind of computation helps in off loading certain tasks to the nearby cloud/cloudlets for execution this makes task scheduling more crucial mutually at both the mobile cloud and the mobile devices. In this research, this crisis have been modelled as a problem of energy consumption optimization problem while considering priority based scheduling, load balancing and reduced power consumption and further solve it by means of Enhanced Bee Colony Optimization based Task Scheduling [EBCO-TS]. A series of iterations were performed to evaluate the recital of the algorithm efficiency and the outcomes are extremely superior and acceptable in contrast to existing methods.

Key words: Energy-efficient, mobile applications, mobile cloud computing, task scheduling, enhanced bee colony optimization based task scheduling algorithm, mobile devices

INTRODUCTION

Mobile Cloud Computing (MCC) integrates cloud computing and wireless communication and attempts to enhance the mobile application's performance hosted at mobile devices like smart phones, PDA's which has been developed rapidly in past few decades. Owing to certain inherent mobile device's defect such as limited battery energy, low CPU speed insufficient storage space and inadequate sensing capacities (Conti *et al*., 2011) mobile applications has numerous confronts in energy management, mobility management, security issues and so on. In order to provide an efficient outcome, MCC succeeds in off loading certain computations have to be performed over certain powerful cloud nodes for instance, cloudlet-which offers higher advantages over conventional mobile services (Kumar and Lu, 2010). With respect to some resource or energy intensive mobile applications hosted in mobile devices, offloading a part to remote cloud saves energy utilization extremely for the devices. Mobile devices send the task to cloud and cloud will execute the job and it will send back the result to corresponding user (Arun and Prabu, 2017a, b) in Fig. 1. Numerous mobile applications like health care, games and e-Commerce are generated in mobile cloud computing concept having it as base (Arun and Prabu, 2017a, b).

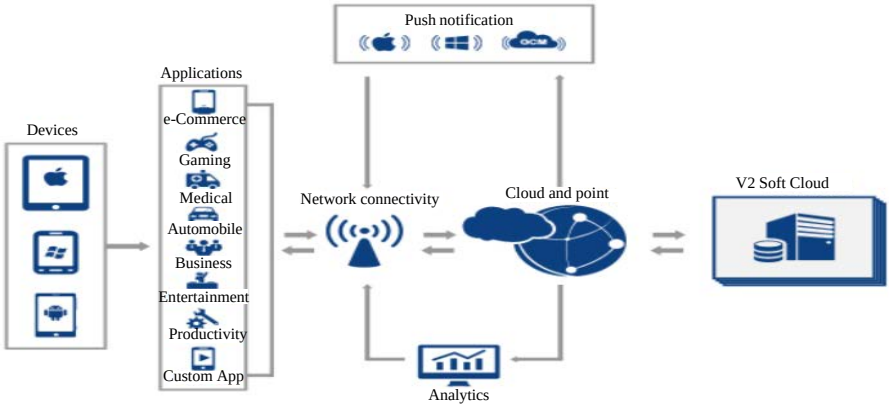

Fig. 1: Two tier architecture of mobile cloud computing

Corresponding Author: C. Arun, Department of Computer Science, Sudharsan College of Arts and Science, Pudukkottai, Tamil Nadu, India

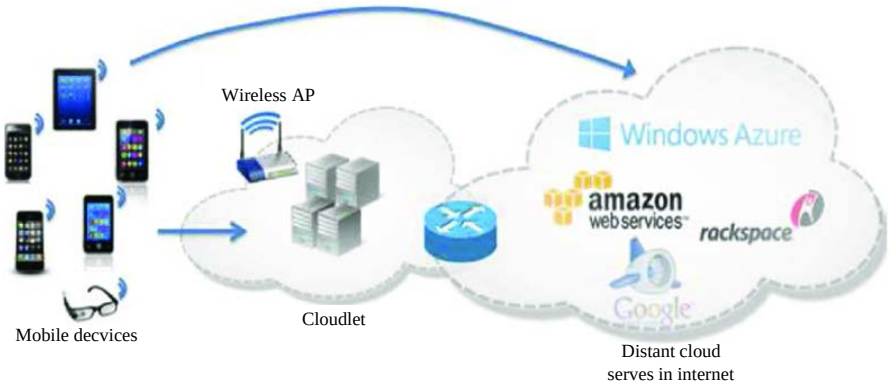

Fig. 2: Offloading model in three tier architecture of mobile cloud computing

Nonetheless, offloading the task is not efficient always as it is influenced by numerous factors like energy consumption of mobile devices during task offloading, wireless channel transmission bandwidth, energy consumption during executing the task at cloud and so on. Concurrently the mobile have to communicate to the cloud for every time it will take more energy for data transmission, so as reduce that energy the new 3 tier mobile-cloudlet-cloud architecture [5] is utilized to bring the cloud nearby to the user. The cloudlet is capable of doing the same thing which is done by cloud in Fig. 2. For example, the inherent attribute of mobile devices are its mobility which stimulates the users to transfer the Access Point (AP) while user travel from one locality to another (Hung *et al*., 2014). This sort of dynamicity leads the wireless connectivity to be unavailable for sometimes. This leads to the elongated waiting time than the usual in addition it leads to refusal of response time for emergency task cases. Therefore, energy consumption is a significant factor which influences the offloading strategy. For instance, if energy consumption due to task offloading at mobile devices and data transmission through wireless channel seems to be huge than the local task execution devoid of offloading (Zhao *et al*., 2013) then there is no use of executing the task remotely, i.e., saving power utilization for mobile devices. Based on numerous states of the art, task scheduling and task offloading encountered in MCC is a multi-objective optimization problem (e.g., execution time, cost, energy), measuring that certain constraints are execution deadline.

For instance for an emergency task, total execution time should not exceed user's specific deadline. Moreover, it is assumed that the tasks acquired from applications are independent which reduces the process of uploading but does not hold MCC environment. For instance, task acquired from application partitioning generally needs certain interactions like data transmission between each other, so as to perform their functionality.

In this research, the task scheduling model was anticipated as a multi-objective optimization crisis with relationship dependency amongst tasks. In specific, every task in the application can either be executed locally over mobile devices or uploaded to the mobile cloud. Subsequently, task that needs continuous interaction with the mobile users are needed to be performed at mobile device, task that consume huge energy and tasks that needs complicated computation are uploaded to the mobile cloud (Wang *et al*., 2017).

Literature review: In this segment, a review over the current works about task scheduling crisis in MCC is discussed. In general, to speed up the application execution time, power consumption or to save storage space, the mobile applications are divided into numerous pieces, termed as tasks in which the tasks are partially scheduled on the execution node of the mobile cloud. The optimization target significantly falls into 2 kinds, either reducing the overall execution time known as makespan or reducing the energy consumption (Lin *et al*., 2015; Guo *et al*., 2016; Tsai *et al*., 2013; Pham *et al*., 2006; Beloglazov *et al*., 2012). As the task scheduling crisis is NP-hard (Wang *et al*., 2017) in which most heuristic methods adopts to solve this crisis which cannot assure to find the optimal outcome but it can acquire the optimal solution.

Chen *et al*. (2015) anticipated a task scheduling method to guarantee enhanced accessibility to cloud network and to rush up processing time in MCC, considering certain constraints like cost and network bandwidth for cloud utilization. Moreover, the methods on how to acquire certain metrics like earliest finishing time or earliest starting time of tasks are not offered in

which the algorithm complexity is unknown. Some of the researches (Wu *et al*., 2013; Razaque *et al*., 2016; Sindhu, 2015) gives more attention to resources in which the nodes in MCC can schedule and process the tasks to the MCC nodes by combining it and the information amount, types of request resource tasks required for execution in order to obtain the more suitable scheduling strategy. Tsai *et al*. (2013) anticipated the task scheduling procedure based on Quality of Service (QoS) metrics, like average execution, load balancing and make span. Initially in accordance to QoS, they compute the priority tasks and then tasks with greater priority are scheduled to the nodes first. Hu *et al*. (2010) anticipated an effectual task scheduling procedure for allocating workflow based on network bandwidth availability. Nonetheless, investigators adopts Min-Min and Min-Max algorithm to adopt tasks to every nodes in cloud based on non-linear programming model.

For tasks acquired from application partitioning, certain tasks are suitable for uploading the MCC or some are not suitable. Selecting the tasks to guarantee and upload the task-precedence requirements and application completion time constraints are fulfilled has acquired huge attention in the past. Ajit and Vidya (2013) describe a strategy which initiated from minimal-delay scheduling strategy and then it performs energy minimization by applying frequency scaling or dynamic voltage technique.

Sindhu (2015) deliberated resource scheduling policy and energy-efficient dynamic offloading to diminish the energy consumption and lessen application execution time, to achieve energy-efficient computation offloading over hard constraint for completion time of application.

The investigators anticipated a Genetic algorithm and a greedy algorithm with an adaptive selection of appropriate mutation and crossover operations to schedule and allocate the real time tasks with heterogeneous virtual machines or precedence constraint. There exists enormous works which dealt with the scheduling and task allocation for real-time works in cloud environment as by Panwar and Mallick (2015), Li *et al*., (2011), Mondal *et al*. (2012). A differential evolution algorithm to schedule tasks set to reduce make span and total cost by integrating Taguchi approach within a framework of differential evolution algorithm to provide enhanced solution to the potential offspring.

Based on few reviews of literature, this research spotlights on reducing the power consumption of mobile devices locally, thus, fulfilling the application completion time constraint and task-precedence requirements by enhanced bee colony optimization for task scheduling. To perform this optimization function, the mathematical theoretical model and the enhanced bee colony optimization for task scheduling algorithm to solve is crisis is anticipated.

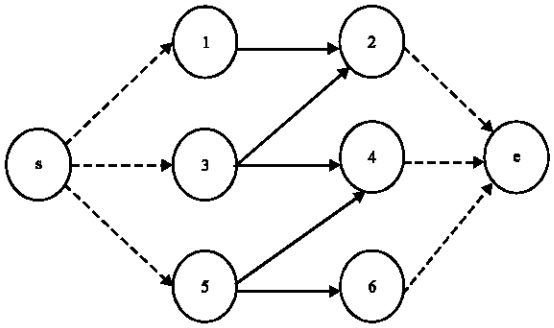

Fig. 3: Pictorial representation of task precedence relationships

System model: Consider a mobile cloud computing environment which preliminarily considers 2 parts: mobile cloud and mobile users correspondingly. The mobile cloud architecture is depicted in Fig. 2 in which the tasks acquired from mobile users are uploaded to cloud through the wireless access point. Therefore, AP offers radio resources and communication support. The mobile cloud performs admission control by monitoring the availability of computing resources like memory, storage and CPU of computing nodes, i.e., Virtual Machines (VMs).

In general, mobile application comprises of task sets in various granularities, we represent the application by a directed Task Graph, TG = (V, E) as in Fig. 2. Every node "i"ϵV in G that specifies an edge and a task e (i, j) that specifies the precedence relationship amongst the tasks "i" and "j" that is j cannot starts its execution until the precedence of task i is completed. Therefore, in task graph there exists only one ending node and starting node correspondingly. The attention given to upload task and scheduling the task has been explained in this research.

A task "t" can be described as a 3-tuple t<= tid it represents the application identification using numerical values, workloads are specified by computation amount when it is executed and Δm specifies the data migration between the tasks with precedence relationships as in Fig. 3.

If certain tasks in the application are involved in uploading the execution, there exist 2 energy consumption strategies at mobile devices. For instance, computational consumption and mobile device needs to utilize energy in executing application's local tasks.

Moreover, the mobile device requires more energy to transmit the tasks remains to the MCC through the wireless communication channels like 3G/4G, WIFI and so on. This type of energy consumption is known as communication consumption. However, if sum of 2 types of consumption is greater than the consumption spent by mobile devices while the complete application is locally executed, thus, there is no point of view to upload tasks to MCC. The energy consumption of mobile devices does not incorporate energy consumption by mobile cloud. For this cause, the MCC does not charge the mobile users. If the costs exceeds over the user's budget, the user can avoid choosing the uploading task to the mobile cloud. Next section illustrates the details of energy consumption and formulates the optimization functionality.

Problem formulation: Assume, the application comprises of "n" tasks. For every task, there exist 2 choices to execute it, i.e., remotely or locally. Let ϕt be the variable to indicate the execution strategy. If Φt = 1, the task "t" will be executed at the mobile device, if it is "0" otherwise. Algorithm 1 contains the notations for task graph precedence.

Algorithm 1; Notations for task graph precedence:

Φt→Variable for indicating execution strategy
s (t)→Computational task size
α_l→Processing speed
T^l_t→Task t execution time at mobile
E^l_t→Energy consumed by mobile in mobile execution
T^c_t→Task t execution time at cloud/cloudlet
E^c_t→Energy consumed by mobile in cloud/cloudlet execution
X_{active}→Power of mobile device during execution
α_C→Processing speed of nodes at cloudlets/cloud
X_{idle}→Idle power
X_{trans}→Transmitting power of mobile device
pred (m)→Set of immediate predecessor asks of task t
RS→Ready state for execution
TT→Termination time of the task
E→Energy utilization
A→bandwidth
T_{app}→Complete execution time
F→The price of computation unit provided by mobile clouds

It the task execution is performed locally, the execution time is computed as follows in Eq. 1:

$$T_t^l = \frac{S(t)}{\alpha_l} \quad (1)$$

where, S (t)→computational task size α_l→processing speed. The energy consumption of mobile device is given as in Eq. 2:

$$E_t^l = \frac{S(t)}{\alpha_l}, X_{active} \quad (2)$$

where, X_{active}→power of mobile device during execution if "t" is executed remotely, execution time can be computed as in Eq. 3:

$$T_t^c = \frac{S(t)}{\alpha_c} \quad (3)$$

where, α_c→processing speed of nodes at cloudlets/cloud. The energy consumption of mobile device is given in Eq. 4:

$$E_t^c = \frac{S(t)}{\alpha_c}.X_{idle} + \frac{S(t)}{A}.X_{trans} \quad (4)$$

Where:
X_{idle}→idle power
X_{trans}→transmitting power of mobile device

While the task is executed in cloud, the mobile device remains in an idle state and X_{idle} is used to compute the mobile device energy consumption. Based on the relationship between the tasks, assume that before scheduling the task "t", all the immediate task predecessor must be completed before the execution for this cause, "t" requires the output results as input parameters. To calculate the complete execution time of application, certain definitions are illustrated.

Def 1 (ready state): The ready state of the task "t" is distinct as the primary starting time, whilst all the immediate predecessor tasks have completed the execution. Therefore, ready state of task "t" is executed locally on mobile device. It is given in Eq. 5:

$$RS_t^l = \max_{pred(m)}\{TT_t^l, \max TT_s^r\} \quad (5)$$

where, pred (m)→set of immediate predecessor tasks of task t, RS→ready state for execution. Time consumption of input and output parameters are ignored while transmission carried out amongst mobile device and cloud. The parameters size is much smaller

than the task itself and the 4G network greater power and WiFi also makes data transmission time neglected. If "s" is executed locally, max $\{Tt_s^l, Tt_s^r\}$, if "s" is scheduled to cloud for execution. Similarly, ready state of task "t" is executed remotely on cloud and it is defined in Eq. 6:

$$RS_t^l = \max\{TT_t^{trans}, \max TT_s^c\} \quad (6)$$

where, TT➔termination time of the task.

Def 2 (Termination time): The termination time of task "t" is given as task "t" time which entirely completes the execution. Thus, termination time of task "t" is executed locally on mobile device and defined in Eq. 7:

$$TT_t^l = RS_t^l + T_t^l \quad (7)$$

Similarly, termination time of task "t" executed remotely on cloud is defined in Eq. 8:

$$TT_t^c = RS_t^c + T_t^c \quad (8)$$

Besides, TT_t^{trans} represent the time when task "t" is completely uploaded to the cloud via. wireless connectivity. The definition is as follows in Eq. 9:

$$T_t^{trans} = \max\{TT_t^c, TT_s^{trans}\} + \frac{S(t)}{A} \quad (9)$$

By these definitions, execution time of entire application is calculated as in Eq. 10:

$$T_{app} = \max\{TT_t^c, TT_t^l\} \quad (10)$$

Assume that while 2 tasks are scheduled with precedence relationships at the same place, communication energy consumption can be avoided. To diminish the mobile device energy consumption, the only way is to off load complete application to cloud. Nonetheless, there are 2 reasons to the mobile users to follow this way. Initially, some applications requires frequent interactions with mobile users like human face recognition which renders communication energy consumptions and even degrade mobile user's quality of service as in Eq. 11. Secondly, difference amongst cloud computing and other computing resources like grid computing lies in cloud computing and earn its own profits. As a result, mobile users enjoy computing convenience by utilizing services offered by mobile cloud. In general, the cost increases as sum of tasks uploaded increases.

$$F = \min E \quad (11)$$

where, E➔Energy utilization

MATERIALS AND METHODS

Designs and scheduling algorithms: It leads to NP-complete problem in case of finding optimal upload decisions for task scheduling. Conversely, heuristic intelligent algorithm is an approach to acquire an optimal solution (Akbari *et al*., 2012). In this research, the Enhanced Bee Colony Optimization for Task Scheduling [EBCO-TS] is used to solve this crisis.

EBCO-TS is stochastic search techniques inspired by the evolution principles and heredity and formally described by Goldberg. It is a robust algorithm for computing NP-hard global optimization incorporating scheduling problems. This algorithm works iteratively to provide better solutions for huge search space. In this research an EBCO based task scheduling to solve optimization problem, of which the description outline of the process is given in algorithm 2.

The anticipated replica reduces the load encountered in cloud environment and improves the cloud performance (Li *et al*., 2011). The anticipated model computes the entire tasks execution time to optimize resource utilization. The algorithm reduces the jobs waiting time in the chain of cloud environment (Karaboga and Akay, 2009).

Algorithm 2; Fitness value computation (Employer bee and scout bee):

Step 1: Parameter initialization
Step 2: Employee bee agent construction using EBCO-TS
Step 3: Local search and fitness evaluation of bee agent
Step 4: I = 0
Step 5: Repeat
Step 6: N = 0
Step 7: Repeat
a. Near Neighbourhood➔cloudlet/cloud search
b. Far Neighbourhood➔/cloudlet/cloud search
c. Fitness value computation
d. Scout Bee Agent assignment to allocate Bee Agent based on mutual probabilities
e. Scout Bee Agent
I. Neighbourhood cloud migration
f. Find best Recruit Bee Agent, replacement of Onlooker Bee Agent
If fitness (Best Recruit Bee Agent) <fitness (Onlooker Bee Agent)
g. Find practical Recruit Bee agent, replace with Best Solution,
If fitness (Best Feasible Onlooker Bee Agent) <fitness (Best)
h. N = N+1;
Step 8: Until (N = Employee Bee Agent)
Step 9: I = I+1;
Step 10: Until (I = Max_iteration)

Algorithm 3; Fitness values computation (Employee forager bee agent and onlooker bee agent):

Step 1: Parameter initialization
Step 2: n←total employee bee agents, M←total onlooker bee agents, I←Maximum Iteration No, α_j←Penalty control parameter
Migration-length←Length of neighbourhood cloud/cloudlet
Step 3: Initialize employee bee agent with EBCO-TS Algorithm
Step 4: Evaluate employee bee agents
Minimization based fitness function
Step 5: Repeat
Cycle = 1
1. No. of Bee Agents = 0, 1 (probability)
2. For every employee bee agents
a. Apply searching strategy of near neighbourhood cloud/cloudlet
If fitness (Near_neighbour)<fitness (Employee forager bee agent) then
Employee forager bee agents = Near_Neighbour
b. Apply search strategy far neighbourhood cloud/cloudlets
If fitness (Far_neighbour)<fitness (Employee forager bee agent) Then
c. Calculate probabilities based on fitness.
e. Identify possible bee agents (onlooker bee agents) that sent to food patches discovered by employee bee agents in accordance to earlier determined probabilities
f. N_i = Number onlooker bee agents send to the i^{th} cloud/cloudlet
g. Onlooker bee agent of ith solution = migration of neighbourhood clouds/cloudlets
h. Determine the fitness significance of each onlooker bee agent, If overall fitness value and bee agents stability (Onlooker bee agents) is enhanced than suitability factor of employee forager bee agent solution then replace with onlooker bee agents solution
5. Best solution
6. Scout bee agents
a.. Bee agents initialization using EBCO-TS Algorithm
b. Total number of worst employee bee agents in total population is compared with bee agent solutions
c. If current bee agent solution is enhanced than employee forager agent solution then replace the recent solution with scout bee agent solution. Else consider employee bee agent solution is reassigned for next round devoid of making further change.
7. Cycle = Cycle+1;
8. Until (Cycle = Total_ iteration)

Algorithm 4; Optimization using local search:

Step 1: Let $S_j = \emptyset \; \forall \; j = 1, ..., m$ (S_j is task given to the agent j)
Step 2: Build separate onlooker bee agents for each task Li., firstly, Li = (1, …, m) $\forall_i$.
Step 3: Consider random order of tasks, I = 1
Step 4: While (no task have been allotted) repeat
a. Select any onlooker bee agent randomly from Li by mutual probability function that rely on bee agent typj and required resource for task i
b. The probability of minimal cost of onlooker bee agent is selected. Assign current task with onlooker bee agent
c. Let I = i+1

Algorithm 5; Neighbourhood cloud/cloudlets:

Step 1: Let S = {i | i? {1,…,n}} , k=1 Migration to neighbourhood = ∂
Step 2: If S = Ø else end; otherwise i_k is ejected
Step 3: Let j* is onlooker agent j which minimizes,
Step 4: Assign i_k to j*, output, calculate fitness
Step 5: If Fitness<fitness (migration to neighbourhood) the migration to neighbourhood
Step 6: k: k+1, return to step 2
Step 7: Output migration to neighbourhood

In the anticipated EBCO-TS algorithm balances load amongst tasks in cloud environment. The algorithm was

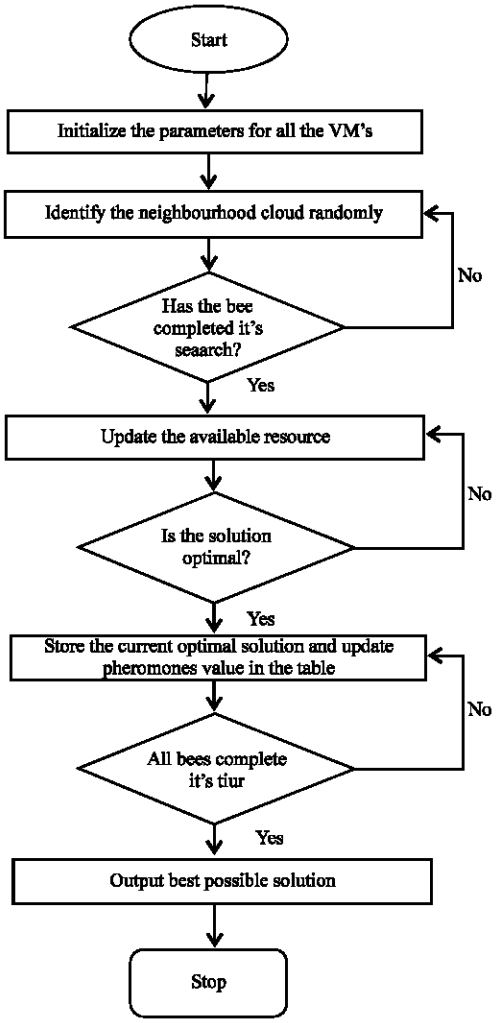

Fig. 4: Flow chart representation for resource allocation strategy based on EBCO-TS algorithm

emerged from bees characteristics that how they acquire food from nest. The specific objective of this research incorporates: the algorithm is better to balances load amongst dependent jobs in cloud computing environment. Figure 4 shows the flow representation of the proposed work. These are numerous steps to overcome the crisis termed cloud load balancing. The procedure is as follows:

- Input parameters of bee's optimization method should be initialized
- Available VMs are measured over cloud environment
- Available VM list is loaded into runtime memory with CPU and RAM
- The list of available resource is revised with available VM list
- Resource capacity is measured for every resource

- Load task list to runtime memory
- Calculate tasks length in earliest finish time to compute each individual task length, subdivide the task and create subtask "i"
- Calculate probability value of available VM
- Compare resource usage of available VMs and task length with probability value
- Evaluate VM Load in resource usage percentage form
- Calculate failure rate of VMs to determine trustworthy VMs to allocate current task
- Shortlist the available VMs which processes the given task
- Assign task "t" to resource with highest probability and minimum response time
- Update resource list with current values of VM load
- Update pheromone value for every available resource
- Repeat initial steps

RESULTS AND DISCUSSION

Simulation and results analysis: This study presents the experimentation through numerical simulations to compute the efficiency and effectiveness of the proposed approach.

Experiment setup: The experiment was performed on the laptop with 8192 M of RAM, 2.5 GHz Intel CPU, Microsoft Window 7 OS. The algorithms are executed in cloud sim environment and computation carried out in diverse parameter settings, like energy utilization, task scheduling and load balancing (Soni and Kalra, 2014) in EBCO-TS. Task graph structure was initialized owing to random simulation. For each task, the workload is generated randomly and data transmitted into subsequent tasks.

In the experiment, once the parameters are initialized for every task, decision to upload it or not are definite. To abridge the system model and eliminate the waiting time of tasks in the uploading process, assume that every task must be uploaded to mobile cloud does not require waiting time (Dasgupta *et al*., 2013). Occasionally, it is essential for enormous calculation through offline method.

While constructing task graph based on real-world application, task graph generally has specific structure, task dependency and number of node relationship. For instance, perform first experimental set based on task graph to verify influence of probability of occurrence and rate of convergence. Figure 5, there are roughly 10 tasks in mobile application, every numerical value over the node represents computational data size of task (Mondal *et al*., 2012) and every numerical value on edge signifies the data migration among 2 tasks with dependency relationships.

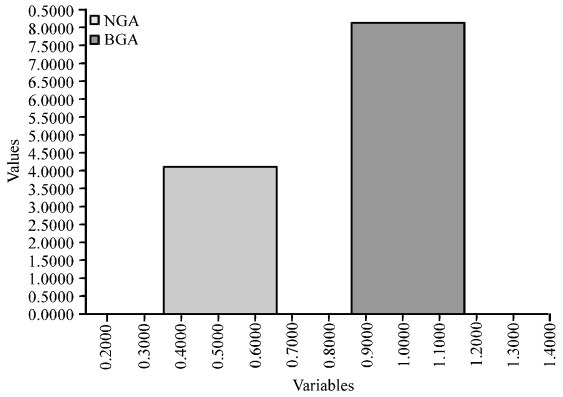

Fig. 5: Representation of task allocation using virtual machine vs. Makespan iterations

The workload of every task and data transmission amongst 2 tasks with dependency relationships are produced randomly.

First, the influence of probability over makespan for finding the optimal solution using EBCO-TS is studied. Based on the parameter settings, first set the probability to default value and Maximum Iteration (MI) to 100. The algorithm runs for 50 times under each probabilistic determination to obtain average makespan and results are depicted in Fig. 5.

Figure 6 shows the graphical representation of task allocation in VM and total number of tasks for experimental iterations. This investigation considers four existing algorithms such as FCFS, Min-Min and PSO along with the proposed EBCO-TS. VM allocates the task appropriately and the proposed method shows superior utilization of resources when compared to the existing methods. Task scheduling is related to the makespan utilization, i.e., ready state and termination time of task. The makespan time of the proposed method is 4.1 sec while the existing methods shows 8.125 sec. Because of this, allocating more number of tasks to the proposed algorithm is higher and efficient. The task scheduled for the proposed method is 50, 52, 54, 56, 58 and 60. An average of additional 2 tasks was scheduled to the proposed EBCO-TS method.

Figure 7 and 8 show the pictorial representation of number of jobs executed in the cloudlet/cloud environment with respective to the execution time. The proposed EBCO-TS shows better trade-off when compared to the existing research. Figure 9 the makespan time utilization is computed for the existing algorithms such as FCFS, Min-Min and PSO with the proposed EBCO-TS method. The stating time and the termination time of the proposed method is performed faster asthe

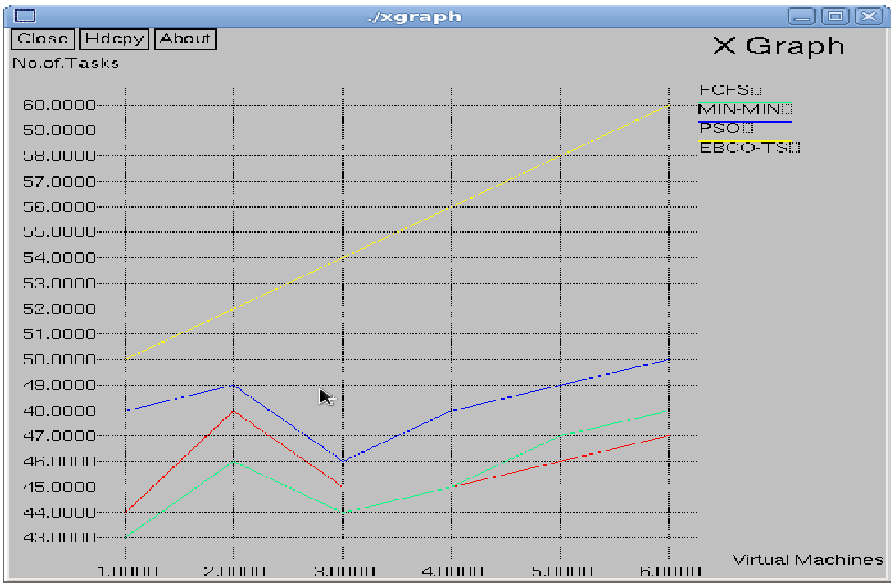

Fig. 6: Graphical representation of task allocation using virtual machine vs. No. of tasks

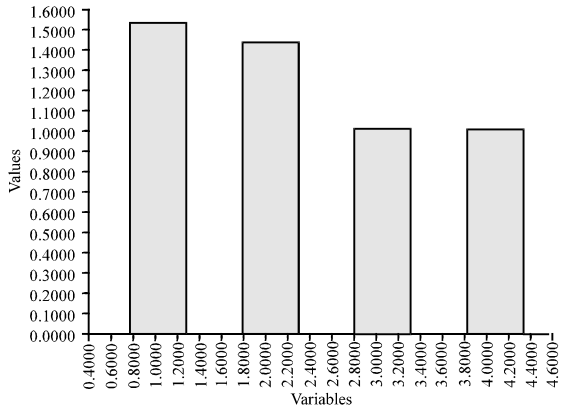

Fig. 7: Graphic all representation of number of jobs running in mobile cloud environment

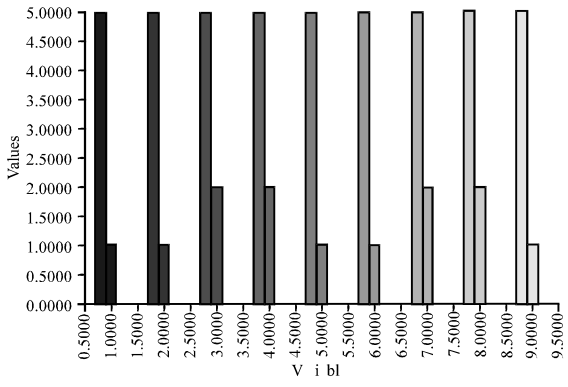

Fig. 8: Execution time for the proposed EBCO-TS algorithm

power consumption of executing a task will be reduced efficiently. The makespan time utilization of the proposed method is 500, 620, 540, 610, 595, 750, 900 sec based on the task allocated. In which it is lesser than that of the existing methods like FCFS, Min-Min, PSO.

The makespan increases when the probability either decreases or increases from 0.02 that is the total time of execution of application reaches minimum (Mondal *et al*., 2012) while the algorithm obtains best solution, i.e., attaining Minimum Energy Consumption (MEC) with probability being 0.02. A striking conclusion is that the average makespan almost increases by 68%, when the probability is set to 0.1 and 0.2, correspondingly. Consequently, it is vital to set suitable probability to solve task scheduling crisis. There are no obvious correlations amongst MEC and iterations when probability varies. When probability falls from 0.02-0.1, average iterations to attain MEC are the same. Therefore, achieving best solution, i.e., minimum energy consumption, typically takes more iterations than ordinary situations. Figure 10 explains the execution time of the proposed EBCO-TS algorithm with the existing research.

The next set of experiments was conducted to verify the probability effect on makespan when attaining MEC as in Fig. 11. The maximum iteration remains 100. It is easier way to attain minimum energy consumption during scheduling (Liu and Wang, 2012) compared to other probability settings. Average make span enlarged by 10% when probability is set to 0.4 and 0.7, correspondingly. It is also decisive to select appropriate probability to resolve task scheduling problem in this research.

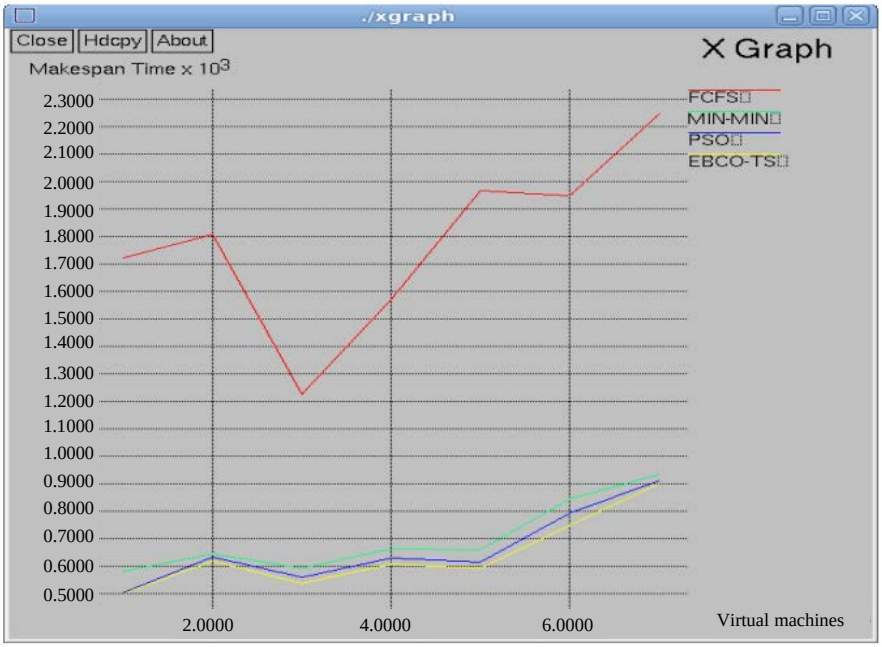

Fig. 9: Graphical representation of virtual machine time vs. Makespan iterations

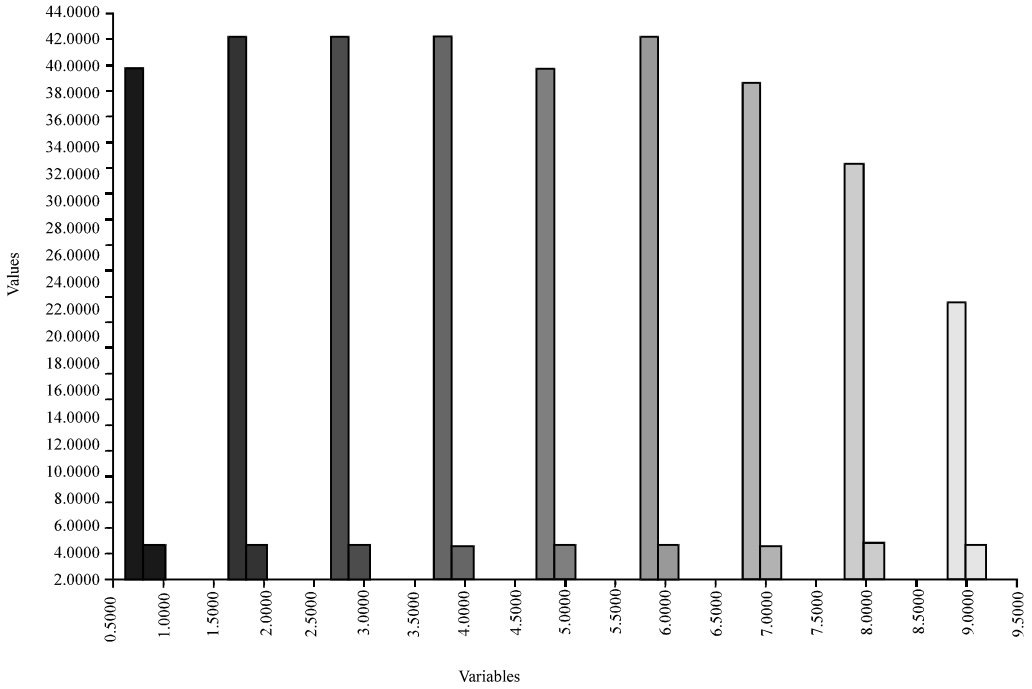

Fig. 10: Representation of total execution time for running a task

Figure 12 and 13, shows the graphical representation of task scheduled based on the percentage of resource utilization and the load balancing strategy while allocating jobs to the cloud/cloudlets. In final experimentation set, generate task graph randomly and number of tasks ranging from 10-12, to verify the effects of tasks on the algorithm running time, when attaining MEC. It is obvious that the amount of tasks have considerable influence over time to obtain best solution. For instance while number of tasks raises from 10-12, corresponding average makespan also increased by 150%. For this reason, execution time of tasks at mobile device or cloud size increases, along

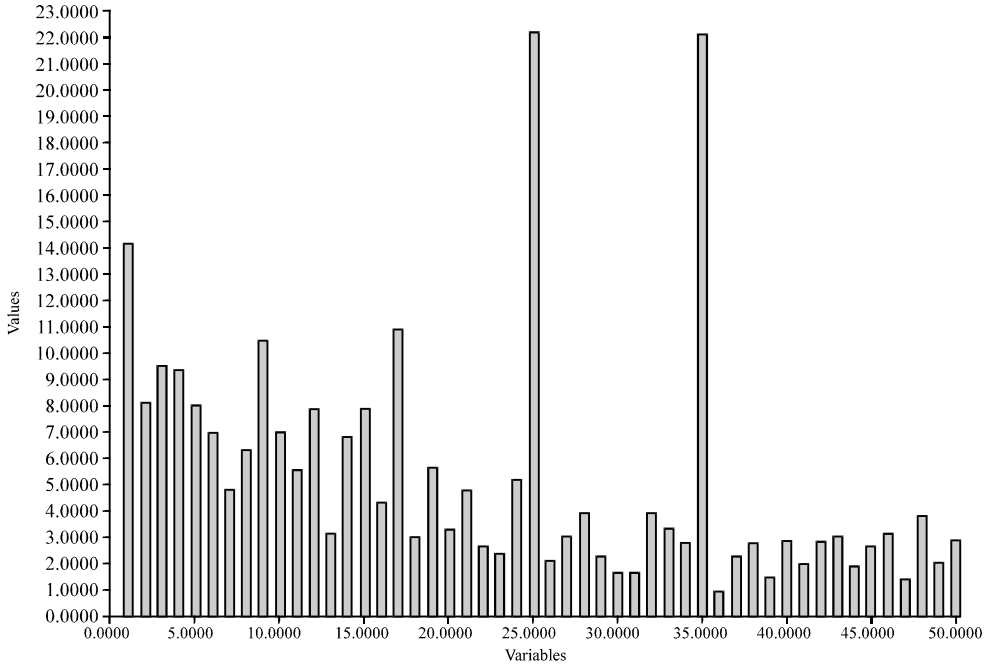

Fig. 11: Representation of task scheduled based on execution time

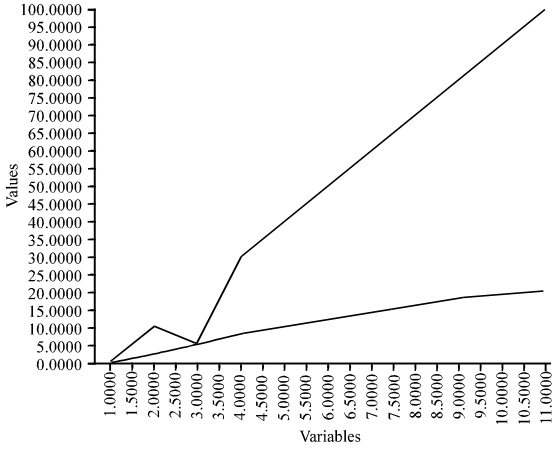

Fig. 12: Graphical representation of energy consumption vs. task allocation

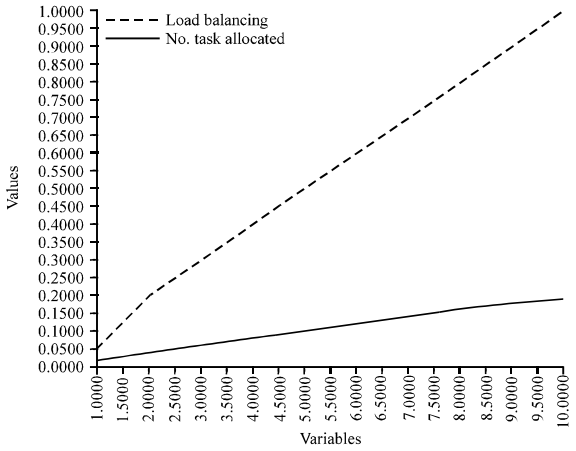

Fig. 13: Graphical representation of allocated task vs. load balancing

with communication time. The number of iterations of EBCO-TS increases, when number of tasks raises. As an outcome, number of iterations also increases.

Figure 14 illustrates the energy consumption of the proposed EBCO-TS with the existing methods like FCFS, Min-Max and PSO. Our proposed method shows a significant change in the energy consumption strategy when various task has been allocated. The anticipated method shows better trade off in comparison with the prevailing works.

Figure 15 shows the load evaluation strategy of the proposed EBCO-TS with the existing algorithms such as FCFS, Min-Min and PSO. The computation shows various changes in the balancing the load when the number of users request for cloudlet increases. The proposed EBCO-TS shows a constant balancing strategy when the number of users increases. Existing algorithms shows lacks in balancing the load, thus, leads to the excessconsumption of energy. The proposed method shoes superior trade off when compared to the existing methods.

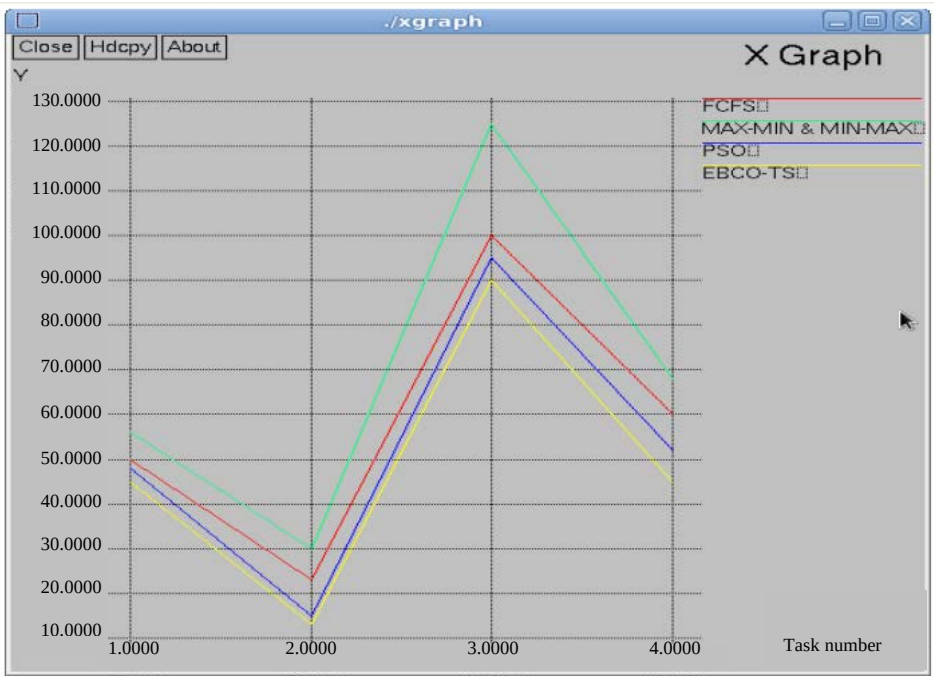

Fig. 14: Existing and proposed system comparison graph for energy consumption

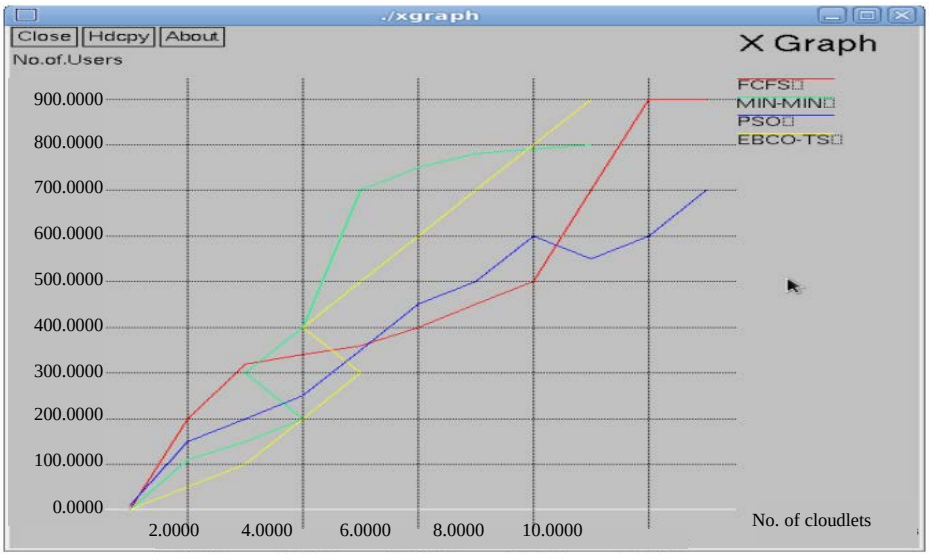

Fig. 15: Graphical representation of load evaluation based number of tasks

CONCLUSION

Scheduling the task in MCC gives an NP-hard problem which has influenced the attention of numerous researchers in past few decades. This research models this crisis as an energy consumption optimization problem while considering data transmission, task dependency, load balancing and some constraint conditions such as ready state and termination time and further solve it using EBCO-TS algorithms. The algorithm shows efficient way offloading the task to cloud and the way to reduce energy consumption. The proposed algorithm shows better trade off compared to the existing methods. The future research can be extended by testing the algorithm's performance with much huge task graphs and develop more proficient heuristic algorithms to resolve the task scheduling crisis.

REFERENCES

Ajit, M. and G. Vidya, 2013. VM level load balancing in cloud environment. Proceedings of the 2013 4th International Conference on Computing, Communications and Networking Technologies (ICCCNT), July 4-6, 2013, IEEE, Tiruchengode, India, ISBN:978-1-4799-3926-8, pp: 1-5.

Akbari, R., R. Hedayatzadeh, K. Ziarati and B. Hassanizadeh, 2012. A multi-objective artificial bee colony algorithm. Swarm Evol. Comput., 2: 39-52.

Arun, C. and K. Prabu, 2017b. Applications of mobile cloud computing: A survey. Proceedings of the 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), June 15-16, 2017, IEEE, Madurai, India, ISBN:978-1-5386-3901-6, pp: 1037-1041.

Arun, C. and K. Prabu, 2017a. Architecture of mobile cloud computing. Pak. J. Biotechnol., 14: 302-304.

Beloglazov, A., J. Abawajy and R. Buyya, 2012. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gen. Comput. Syst., 28: 755-768.

Chen, H., X. Zhu, H. Guo, J. Zhu and X. Qin *et al*., 2015. Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment. J. Syst. Software, 99: 20-35.

Conti, M., S. Chong, S. Fdida, W. Jia and H. Karl *et al*., 2011. Research challenges towards the future internet. Comput. Commun., 34: 2115-2134.

Dasgupta, K., B. Mandal, P. Dutta, J.K. Mandal and S. Dam, 2013. A Genetic Algorithm (GA) based load balancing strategy for cloud computing. Procedia Technol., 10: 340-347.

Guo, S., B. Xiao, Y. Yang and Y. Yang, 2016. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. Proceedings of the 35th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM 2016), April 10-14, 2016, IEEE, San Francisco, California, ISBN:978-1-4673-9954-8, pp: 1-9.

Hu, J., J. Gu, G. Sun and T. Zhao, 2010. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming, December 18-20, 2010, Dalian, China, pp: 89-96.

Hung, P.P., T.A. Bui and E.N. Huh, 2014. A New Approach for Task Scheduling Optimization in Mobile Cloud Computing. In: Frontier and Innovation in Future Computing and Communications, Park, J.J., A. Zomaya, H.Y. Jeong and M. Obaidat (Eds.). Springer, Dordrecht, Netherlands, ISBN:978-94-017-8797-0, pp: 211-220.

Karaboga, D. and B. Akay, 2009. A comparative study of artificial bee colony algorithm. Applied Math. Comput., 214: 108-132.

Kumar, K. and Y.H. Lu, 2010. Cloud computing for mobile users: Can offloading computation save energy? Computer, 4: 51-56.

Li, K., G. Xu, G. Zhao, Y. Dong and D. Wang, 2011. Cloud task scheduling based on load balancing ant colony optimization. Proceedings of the 2011 6th Annual International Conference on China Grid, August 22-23, 2011, IEEE., Dalian, Liaoning China, pp: 3-9.

Lin, X., Y. Wang, Q. Xie and M. Pedram, 2015. Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. IEEE. Trans. Serv. Comput., 8: 175-186.

Liu, Z. and X. Wang, 2012. A PSO-based algorithm for load balancing in virtual machines of cloud computing environment. Proceedings of the International Conference on Swarm Intelligence, June 17-20, 2012, Springer, Berlin, Heidelberg, Germany, ISBN:978-3-642-30975-5, pp: 142-147.

Mondal, B., K. Dasgupta and P. Dutta, 2012. Load balancing in cloud computing using stochastic hill climbing-a soft computing approach. Procedia Technol., 4: 783-789.

Panwar, R. and B. Mallick, 2015. Load balancing in cloud computing using dynamic load management algorithm. Proceedings of the 2015 International Conference on Green Computing and Internet of Things (ICGCIoT'15), October 8-10, 2015, IEEE, Delhi, India, ISBN:978-1-4673-7909-0, pp: 773-778.

Pham, D.T., A. Ghanbarzadeh, E. Koc, S. Otri and S. Rahim *et al*., 2006. The bees algorithm-a novel tool for complex optimisation problems. Proceedings of the 2nd I*PROMS Virtual International Conference on Intelligent Production Machines and Systems, July 3-14, 2006, Elsevier, ýAmsterdam, Netherlands, pp: 454-459.

Razaque, A., N.R. Vennapusa, N. Soni and G.S. Janapati, 2016. Task scheduling in cloud computing. Proceedings of the 2016 IEEE Conference on Long Island Systems, Applications and Technology (LISAT), April 29, 2016, IEEE, Farmingdale, Long Island, New York, USA., ISBN:978-1-4673-8490-2, pp: 1-5.

Sindhu, S., 2015. Task scheduling in cloud computing. Intl. J. Adv. Res. Comput. Eng. Technol., 4: 3019-3023.

Soni, G. and M. Kalra, 2014. A novel approach for load balancing in cloud data center. Proceedings of the 2014 IEEE International Conference on Advance Computing (IACC), February 21-22, 2014, IEEE, Gurgaon, India, ISBN:978-1-4799-2572-8, pp: 807-812.

Tsai, J.T., J.C. Fang and J.H. Chou, 2013. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. Comput. Oper. Res., 40: 3045-3055.

Wang, J., J. Tang, G. Xue and D. Yang, 2017. Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems. Comput. Networks, 115: 100-109.

Wu, X., M. Deng, R. Zhang, B. Zeng and S. Zhou, 2013. A task scheduling algorithm based on QoS-driven in cloud computing. Procedia Comput. Sci., 17: 1162-1169.

Xia, F., F. Ding, J. Li, X. Kong and L.T. Yang *et al*., 2014. Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing. Inf. Syst. Front., 16: 95-111.

Zhao Y., L. Chen, Y. Li, P. Liu and X. Li *et al*., 2013. RAS: A Task Scheduling Algorithm Based on Resource Attribute Selection in a Task Scheduling Framework. In: Internet and Distributed Computing Systems, Pathan M., G. Wei and G. Fortino (Eds.). Springer, Berlin, Heidelberg, ISBN:978-3-642-41427-5, pp: 106-119.