Received: November 9, 2016

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Evolutionary Multi-Objective Approach for Resource Scheduling in Mobile Cloud Computing

Dasari Nagaraju[1] **Vankadara Saritha**[1*]

[1]*VIT University, Vellore, Tamilnadu, India*
* Corresponding author's Email: vsaritha@vit.ac.in

Abstract: Mobile cloud computing (MCC) is one of the evolving fields in recent years. The complexity of MCC made researchers to concentrate on efficient application development. In MCC, resource scheduling is treated as one of the major issues. Genetic Algorithms (GAs) are efficient search techniques to find the optimal solution for the scheduling problem. GAs has the ability to optimize the resource scheduling in both homogeneous and heterogeneous environments. This paper presents the multi objective genetic algorithm for MCC (MOGAMCC) environment. To implement the MOGAMCC, the cloudsim toolkit was extended with the MOGA and its task scheduling approach determines the optimal scheduling policy. A single point crossover model is employed for the generation of new population. Mutation process is carried by randomly changing the bit positions in the chromosomes. The experimental results show that the proposed model finds the optimal trade-off between the defined objectives and which ultimately reduces the makespan.

Keywords: Directed acyclic graph, Genetic algorithm, Mobile cloud computing, Resource scheduling.

1. Introduction

In recent years, mobile cloud computing (MCC) has gained popularity from both academic and industry. The MCC is reducing the complexity of applications and increasing the performance of the mobile devices. With the advanced development in the capabilities of the mobile devices such as network capacity, CPU power and sensors, mobile devices are becoming more and more popular and they are access to the internet from anywhere and anytime. The overall internet users in India have exceeded 375 million, accounted for 30% net users [1]. This will create a lot of opportunities for the growth of mobile applications such as navigation, speech recognition, online gaming and online streaming. But, the limitation of mobile devices like limited battery power, computation speed and memory restricting the application developers to execute the complex applications.

To overcome these limitations, MCC is introduced to offload the computational task to the cloud. The cloud offering service providers are iCloud and Amazon EC2. Cloud is capable of providing any type of services to the end users. Cloud was supported by virtual machine technology, and it is helpful to provide services extensively and elastically. K. Kumar et al. proposed an effective approach to the task offloading in MCC. This approach concentrated on the process of reducing the energy utilization of mobile devices, which investigates the offloading of jobs to improve the efficiency of the battery in mobile devices [2]. Z. Li et al. introduced a method for computational offloading to reduce the power for mobile devices. The mobile devices offload the applications to remote server for computation [3-4]. However, the application partitioning for offloading is a problem and consider it as an NP-Complete [5]. Shumao et al. [6] proposed a mechanism for effective offloading scheme using middleware platform that can provide job offloading to the mobile devices. For instance, fuzzy control based inference engine for offloading was proposed by X. Gu et al. The main theme of the algorithm is to reduce the networking cost between the wireless devices and remote servers [7]. The algorithm partitions the application and uploads to the nearby remote servers for execution [20].

However, these algorithms are failed to explain the role of response time in the offloading process for both mobile devices and remote servers where an appropriate explanation about the time constraint is needed in both wireless and remote environment [21]. The MCC is implemented in various fields such as disaster recovery, image processing and natural language processing [8]. The mobile devices can be used by users in different ways. Allocation of the cloud services to the mobile devices may vary based on the application and it will result in under or over utilization. Therefore, the allocation of the resources to the mobile users must be in dynamic nature.

In this paper, the proposed model addresses the resource constraint issue in mobile cloud computing. The mobile devices are not rich in resources; therefore proper allocation of cloud resources to the mobile devices is necessary to achieve the throughput of the application. In this context, the multi objective resource scheduling mechanism is proposed. The novelty of the proposed algorithm is it considers four major objective functions for task scheduling. The proposed evolutionary algorithm had major advantage in reducing the data transfer time, minimizing the queue length and increasing the resource utilization while comparing with the existing algorithms.

The rest of the paper is organized as follows. Section 2 describes about the related work regarding the recent trends in the field of optimization in the mobile cloud. Section 3 explains about the environment of mobile cloud and the objective functions of the proposed model. Section 4 frames the problem formulation for resource scheduling. Section 5 provides a complete explanation about the multi-objective genetic algorithm for mobile cloud computing. Section 6 discuss about the simulation environment and the results evaluation. Finally, Section 7 concludes the overall contribution of the research work.

2. Related Work

There are a number of scheduling approaches to solve the relatively routine problems in mobile cloud computing. The major research issue in MCC is resource scheduling and it has direct impact on the service cost and response time. Therefore, the usage of optimization algorithms for reducing the runtime of the tasks increases the effectiveness of the solution. In [9], the author proposed modified heuristic approach for task scheduling in cloud. The objectives considered for the optimization are maximization of CPU capacity and effective utilization of the available resources. The drawback of the suggested method is energy consumption of the resources. In [10], Tayal proposed an optimization approach for optimizing the job scheduling process, which is called as Fuzzy-GA based optimization. This model considers the scheduling decision by computing the cluster of tasks. But, it will affect the overall completion time of the tasks. Li et al. [11] proposed ant colony optimization algorithm for finding the optimal scheduling results. This method considers the makespan and mean task completion time as the objective functions. The introduced method is not able to address the queue length of the virtual machines. A multi objective algorithm was proposed by Jahnke et al. [12] for work flow scheduling in cloud computing. The algorithm minimizes the total execution time and cost by employing the pareto model. The pareto model increases the data transfer time and it is a major drawback of the model. In [13], the authors proposed the PSO model for multi objective resource scheduling. They consider the optimizations function as the minimization of cost and task completion time. These proposed strategies are only concentrated on the cost and makespan of the tasks, but neglected the resource utilization factor.

Meanwhile, there are numerous studies has been supported in the reduction of power consumption using optimization algorithms. In [14], the authors proposed an energy aware scheduling mechanism for tasks in multi core systems. The algorithm uses the integer programming model for finding the optimal scheduling of tasks in the multi-cores. They concentrated only on energy consumption and neglected the resource utilization issue. In [15], Wang et al. proposed an energy aware scheduling model for tasks in cloud. As an initial step, it calculates the energy consumption of the cloud and then adjusts the data based on the network states, and finally an integer bi-level programming model was developed for task scheduling using data patterns. The bi-level model will alone be not sufficient to identify the resource capacity for task allocation.

3. MCC Environment

3.1 Task Model in Mobile Environment

The mobile environment is where a wireless device is connected to the cloud in ad hoc manner.

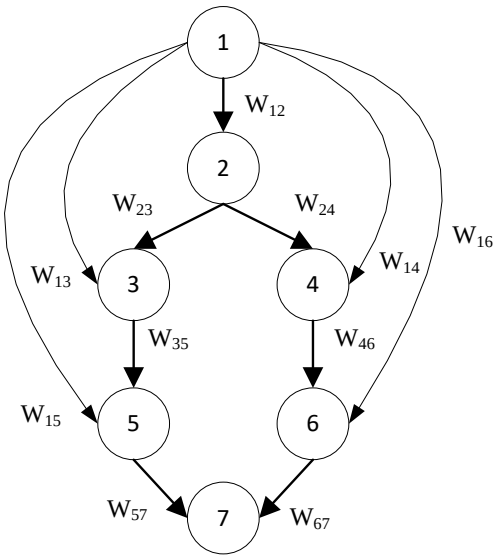

Figure.1 Directed Acyclic Graph for 7 subtasks

Whenever the user submits the job to the mobile environment, the job is partitioned into subtasks for parallel execution. The mobile environment has some limitations i.e., battery power, computation capacity, etc. The task offloading process is made used to overcome these limitations. The mobile environment follows the properties of directed acyclic graph for task submission. The DAG is represented as $G = \{V, E\}$, where V is the submitted job contains set of tasks $T = \{T_1, T_2 \ldots T_x\}$ and E is the set of connection between any two tasks, T_i and T_j. E is formulated as $\{(T_i, T_j, W_{ij}) | i \neq j\}$, where W_{ij} is the weight of the edge. i.e. data transferred from T_i to T_j. The example for the workflow of tasks as DAG is given in Figure.1.

3.2 Cloud Resource Management

In this paper, the proposed model considers the heterogeneous VMs comprising various combinations of memory, CPU and bandwidth. The CPU of the VM decides the completion time of the task and network bandwidth decides the data transmission time. The complete organization of mobile cloud computing is shown in Figure.2. Cloud is a combination of data centers, each data center is a combination of servers, each server has different Virtual machines and each virtual machine is configured with number of processing units called as CPUs. The mobile clients submit the tasks to the cloud for computation purposes and the scheduling of tasks is done based on the MOGAMCC.

Table.1 Definition of terms

Symbol	Explanation
D	Data transfer time of the tasks
S	Represents the starting time of the task
F	Represents the finishing time of the task
C	Represents the chromosome
$W(T_i)$	Amount of data that task T_i is assigned
W_{ij}	Data transfer from task T_i to task T_j
BW	Network bandwidth
$CT(T_i)$	Completion time of task T_i
$CC(VM_k)$	Computation capacity of the virtual machine k
$O(VM_k^{mem})$	Allocated memory at the time of initialization of virtual machine VM_k
$O(VM_k^{cpu})$	Allocated number of CPUs at the time of initialization of virtual machine VM_k
$A(T_j^{mem})$	Assigned memory to the task T_j
$A(Tjcpu)$	Assigned number of CPUs to the task T_j
$R(VM_k^{mem})$	The remaining memory of the virtual machine VM_k
$R(VM_k^{cpu})$	The remaining number of CPUs in the virtual machine VM_k

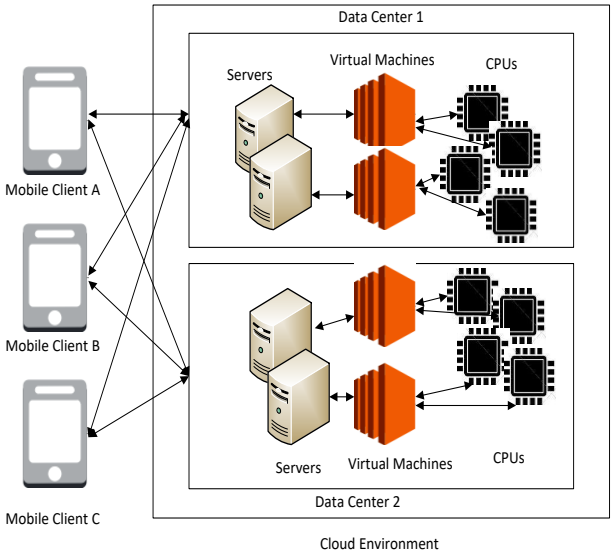

Figure.2 Mobile Cloud Computing Environment

3.2.1. Task Completion Time

The completion time of the task T_i depends on the allocated virtual machine. The allocated CPU and the memory decide the computation capacity of

the virtual machine. Let us assume that, no task is previously assigned to the virtual machine VM_k, T_i is the only task executing on the virtual machine VM_k, then the completion time CT of the task T_i is given as follows.

$$CT(T_i) = W(T_i) \times CC(VM_k)^{-1} \quad (1)$$

Where $W(T_i)$ represents the amount of data assigned to the virtual machine k by the task T_i and $CC(VM_k)$ represents the computation capacity of the virtual machine VM_k.

3.2.2. Data Transfer Time

The network bandwidth is usually different for different virtual machines. Therefore, the virtual machines which having the higher computation capacity usually have the higher bandwidth. The data transfer time D in between task T_i and T_j is given as

$$D(T_i, T_j) = W_{ij} \times (BW(VM_k, VM_p))^{-1} \quad (2)$$

Where VM_k and VM_p are the two types of virtual machines to which tasks T_i and T_j are scheduled, BW is the allocated bandwidth and W_{ij} represents the amount of data transferred from task T_i to task T_j. The Data transfer time is equal to 0, if the tasks T_i and T_j are executing on the same virtual machine otherwise the computation is carried by using equation 2. The objective function for the data transfer time is formulated as

$$f_1 = \sum_{i=1}^{x} \sum_{j=1}^{x} D(T_i, T_j) \quad (3)$$

3.2.3. Length of Task Queue

To optimize the scheduling of the MCC, the length of the task Queue is considered as a one of the objective function. The Queue length of the VM is increased when the assigned tasks to the VM is more than its CPUs, so to achieve the optimization in scheduling, the queue length of the VM has to be minimized. This will reduce the makespan of the proposed model. The $QL(VM_k)$ controls the length of the task queue of VM_k by calculating the remaining memory $R(VM_k^{mem})$ and the remaining number of CPUs $R(VM_k^{cpu})$.

$$QL(VM_k) = \frac{W(T_i)}{R(VM_k^{mem}) \times R(VM_k^{CPU})} \quad (4)$$

Where,

$$R(VM_k^{mem}) = O(VM_k^{mem}) - A(T_j^{mem}) \quad (5)$$

$$R(VM_k^{cpu}) = O(VM_k^{cpu}) - A(T_j^{cpu}) \quad (6)$$

Here, $O(VM_k^{mem})$ represents the allocated memory at the time of initialization of virtual machine VM_k, $A(T_j^{mem})$ represents the assigned memory from the virtual machine VM_k to the task T_j. $O(VM_k^{cpu})$ represents the allocated number of CPUs at the time of initialization of virtual machine VM_k, $A(T_j^{cpu})$ represents the assigned CPUs from the virtual machine VM_k to the task T_j.

The objective function for the length of the task queue is given as

$$f_2 = \sum_{k=1}^{n} QL(VM_k) \quad (7)$$

3.2.4. Makespan

The makespan of the proposed system is calculated based on the finishing time of the task T_{end}. The two functions such as start time and finish time of the scheduled tasks must be calculated. The starting time of the task T_i will depend on the finishing time of the predecessor task T_j.

$$S(T_{initial}) = 0 \quad (8)$$

$$S(T_i) = F(T_j) + D(T_j, T_i) \quad (9)$$

$$F(T_i) = S(T_i) + C(T_i) \quad (10)$$

Where $S(T_{initial})$ represents the starting time of the task $T_{initial}$, $F(T_i)$ represents the finishing time of the task T_i. The objective function for the makespan is given as

$$f_3 = FTime(T_i) \quad (11)$$

4. Problem Formulation

This section describes about the multi-objective task scheduling model based on the predefined objective functions. These objective functions are used to minimize the data transfer time, length of the task queue and makespan.

Problem: $F = \min\{f_1, f_2 \text{ and } f_3\}$

Subject to $\forall i, j = 1,2,..x$, $\forall k = 1,2,..n$. (12)

5. Multi Objective Genetic Algorithm for Scheduling in MCC (MOGAMCC)

MOGAMCC is used for efficient resource scheduling by minimizing the objective functions. The genetic algorithm has the following steps: population generation, selection, crossover and mutation.

As an initial step, N number of population is selected randomly. Then apply the selection, crossover and mutation to produce the first generation child population. Then, for the second generation population, the first generation child population is merged with the parent population and sorted. This process will continue until it reaches to the stopping criteria

5.1 Chromosome Representation

In genetic algorithm, the chromosome C represented by a series of tasks and each task is associated with the virtual machine as shown below.

$$C = T_i(VM_k) \quad (13)$$

Where $i = 1,2,3\ldots x$ *and* $k = 1,2,3\ldots n$. Figure. 3 show the chromosome representation consisting of 7 tasks and 4 virtual machines.

5.2 Initialization of the Population

The major step in the genetic algorithm is population initialization, the good initialization leads to the best solutions in the search space and otherwise the algorithm leads to the bad solutions. Here, the proposed meta-optimization model follows the eq. 1 for calculating the completion time of the tasks. If the count of the tasks is equal to the number of VMs, then the VMs are assigned randomly to the tasks. The newly arrived tasks in the cloud are assigned to the VMs based on the $R(VM_k^{mem})$ and $R(VM_k^{cpu})$.

Algorithm 1: Population Initialization

$T = \{T_1, T_2, \ldots T_x\}$
$VM = \{VM_1, VM_2 \ldots VM_n\}$
Begin
1. For i = 1 to x do
2. For k = 1 to n do
3. If ($i \leq k$) then
4. Compute completion time of task T_i from eq. 1.
5. Select the virtual machine VM_k to the task T_i randomly.
6. Assign the task T_i to VM_k
7. Remove the task T_i from the task set T
8. If T_i finishes its execution on VM_k then
9. Add VM_k to the virtual machine set VM
10. End if
11. End if
12. Otherwise
13. Compute $R(VM_k^{mem})$ and $R(VM_k^{cpu})$ of virtual machine k from eq. 5 and eq. 6
14. If ($(R(VM_k^{mem})=0)$ *or* $(R(VM_k^{cpu})=0)$ then
15. Delete VM_k from the virtual machine set VM
16. Otherwise
17. Find the suitable virtual machine VM_k which will finish the T_i earliest.
18. Assign the task T_i to VM_k
19. Remove the task T_i from the task set T
20. If T_i finishes its execution on VM_k then
21. Add VM_k to the virtual machine set VM
22. End if
23. End for
24. End for
End

Algorithm 2 shows the complete procedure of multi-objective genetic algorithm for MCC. The calculation of fitness function for the randomly selected population is done by the evaluation phase and it is shown in the next section.

Algorithm 2: Multi-Objective Genetic Algorithm for MCC

Generate initial population using the algorithm 1

Repeat
Calculate the fitness function of the chromosomes by using eq. 3, eq. 7 and eq. 11.
Generate the offspring production by performing crossover and mutation.
Replace the parent chromosome with newly generated child population.
Until convergence criteria must be met

5.3 Evaluation Phase

In the evaluation phase, fitness function decides the quality of the schedule. In the proposed method, it is considered three objectives as the fitness functions such as data transfer time, the length of the task queue and makespan which is given in eq. 3, 7 and 11. The main goal is to minimize the data transfer time, task queue and the makespan through

an intelligent scheduling mechanism. After the evaluation phase the selection phase is initialized.

5.4 Roulette Wheel Selection Method

The roulette wheel selection method finds the optimal choice based on the winning probability from the given options [16]. The initial population is selected by using the algorithm 1 and for the next generation of the population is selected by r number of random experiments, i.e. the probability of selecting the chromosome C_l from the pool $P=\{C_1, C_2, \ldots C_m\}$ is dependent on the following eq.14.

$$sel(C_l) = Fitness(C_l) \times (Fitness(P))^{-1} \quad (14)$$

The roulette wheel consists of slices of chromosomes which is directly proportional to the fitness of the chromosome. The selection of a chromosome is done based on the spin of the wheel. An example for roulette wheel process is given in Figure 4. A complete roulette wheel represents all chromosomes in the population, the size of the chromosome depends on the fitness function and for producing N individuals, the wheel has to spin for n times. For instance, in Figure.4 the chromosome 1 has 30 % of chances for selection and chromosome 4 has 4% of chances. Therefore chromosome 1 is selected as optimal solution. Finally, the roulette wheel selection determines which individuals and how many of them can be kept in the next generation.

5.5 Crossover Process

In the task scheduling process, the scheduling mechanism follows the dependencies exist between the tasks. For instance, task T_i is a successor of task T_j, then T_j should execute first in the scheduling process. For an understanding purpose, the proposed model considers the Figure.1 for task dependencies. The crossover process follows the procedure of exchanging the genes between two chromosomes. The popular method is a single point crossover between two parent populations. The cross over rate lies in the range of 0.6 to 0.8. In Figure.5, it is shown that two chromosomes C_1 and C_2 are selected for the generation of new population.

1	2	3	4	5	6	7	Tasks
3	2	3	4	1	2	1	VMs

Figure.3 Chromosome Representations

For producing new child chromosome C_3, first three places are identical to the C_1 and the second part is obtained by reordering of C_1 according to sorting order defined by C_2. The same process is repeated for the generation of C_4.

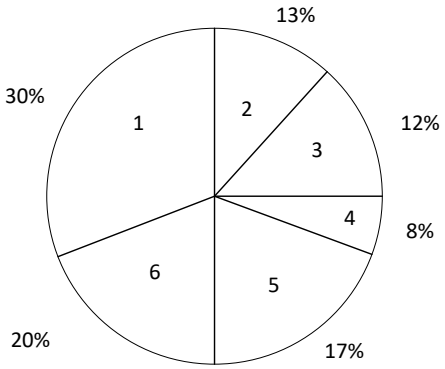

Figure.4 Roulette wheel selection

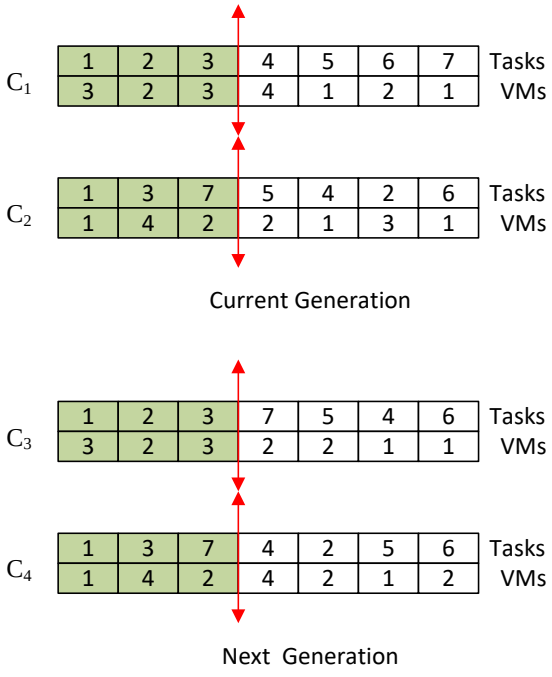

Figure.5 Crossover operation in the multi-objective genetic algorithm

5.6 Mutation Process

Mutation process is identical to the biological process and it is used to produce the diverse population of chromosomes for the new generations. This mutation process is carried by randomly changing the bit positions in the chromosomes. The probability of the mutation rate must be low otherwise it leads to the re-initialisation of the population. The mutation process (Figure. 6) is carried in the following way:

Step 1: As an initial step, a random position is selected for mutation (from Figure. 6, it is Task 5 in C_1).
Step 2: Identify the first parent to the left of the chosen task (from Figure. 6, it is Task 1 in C_1)
Step 3: Identify the first child to the right of the chosen task (from Figure. 6, it is Task 6 in C_1)
Step 4: This can define the range of the chosen task to be shifted by maintaining the sorted order (from Fig. 6, it is identified as 1 to 6 positions).
Step 5: Choose the random position for a new location (for example, the position is selected as 3).
Step 6: Then, Task 5 is relocated to the position 3 (both Task and VM are relocated).

Matching mutation is performed by selecting the VM position randomly and modifies the position of VM by assigning another VM randomly.

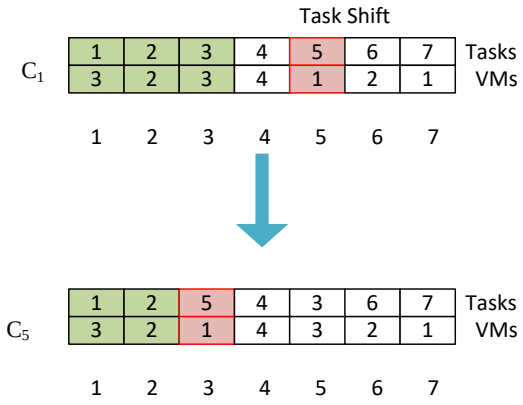

Figure.6 Mutation operation in multi-objective genetic algorithm

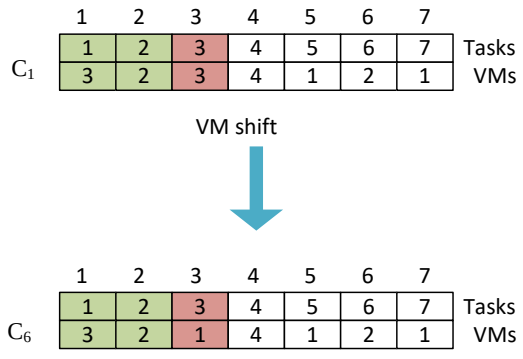

Figure.7 Matching Mutation process for multi-objective genetic algorithm

For instance, in Figure.7 the random position of VM is chosen as 3 and the VM_3 which is replaced with VM_1.

5.7 Analysis of Time Complexity

The time complexity of the initialization phase is O(x), where x is the number of tasks. The time complexity of the crossover process is $O(x^2)$. The time complexity in evaluation process for each chromosome has O(3e), where e is the fitness function for each chromosome. The mutation process has the time complexity of O(x). The time complexity of MOGAMCC is given as O(Q,x), where Q defines the maximum iterations.

6. Results and Discussion

The simulation of the MOGAMCC is carried by using the simulator called as cloudsim toolkit [16]. To implement the proposed algorithm, the cloudsim toolkit is extended with the NSGA-II package [19] for optimizing the task scheduling. The DatacenterBroker class having the bindcloudletToVm() which is responsible for allocating the tasks to the VMs. The cloud simulator works with three inputs: cloud settings, workflow traces and task description. To create the network between the servers, the network topology is implemented by using the predefined network properties of CloudSim. This network topology creates the random network model which is analogous to the internet. The details of the job such as a number of tasks, the number of VMs and the computation capacity are required to initialize the simulator. The initialization of the properties of task and VMs were randomly distributed among the predefined set of VMs and task parameters. Table 2 and Table 3 show the configuration of cloud for experimental evaluation. The data transfer time (DTT), Length of the task queue (QL) and makespan are the selected objectives for the MOGAMCC and these objectives are given as the fitness function for the evaluation of chromosomes in the MOGAMCC.

6.1 Results Evaluation

The evaluation of the MOGAMCC is carried with the other existing optimization methods such as Min-Min [18] and MOSA [17]. The three objective functions are considered for evaluation of proposed MOGAMCC i.e. Data Transfer Time, Queue Length and makespan.

Table.2 VM Configuration Setup

IDs	Memory(KB)	MIPS	Bandwidth	CPUs	VMM name
1-5	512	500	1000	4	Xen
6-10	256	300	10000	1	Xen
11-15	512	500	1000	1	Xen
16-20	512	200	1000	2	Xen
21-25	256	500	10000	1	Xen

Table.3 Task Characteristics

IDs	File size (MB)	Length	Required CPUs
1-50	250	25000	1
51-100	300	45000	1
100-150	300	45000	1
151-200	300	45000	1
201-250	250	25000	1
250-300	250	25000	1

Table.4 Parameters for MOGAMCC

Parameter	Value
Maximum iterations	500
Crossover rate	0.8
Mutation rate	0.1
Size of the population	300
Convergence criteria	20 generations

Figure.8 shows the variation of Data Transfer time for different optimization methods. The min-min algorithm doesn't consider any mechanism for reducing the data transfer time between the tasks, but the MOSA and MOGAMCC had the efficient mechanism for minimizing the data transfer time. The proposed algorithm records 14% improvement while comparing with the Min-Min and MOSA. The MOSA has 4% improvement while comparing with the Min-Min. So, it is observed that the MOGAMCC performs well in reducing the data transfer time.

The plots for Queue Length and makespan obtained from the MOSA, Min-Min and MOGAMCC are illustrated in the Figure.9 and Figure.10. The queue length of the MOGAMCC is calculated by eq. 7. The $QL(VM_k)$ controls the length of the task queue of VM_k by calculating the remaining memory and the number of CPUs. The $QL(VM_k)$ controls the allocation of multiple tasks to the single VM. In Figure.9, it is identified that the value of Queue Length of MOGAMCC is reduced when compared to the Min-Min and MOSA. This is due to the effective optimization mechanism applied by the MOGAMCC. The major drawback of Min-Min algorithm is it has poor load balancing mechanism while comparing with the proposed algorithm. The MOGAMCC utilizes the round robin policy to schedule the task to the available VMs. Therefore, MOGAMCC had better performance in terms of Queue length against the MOSA and Min-Min. Figure.10 illustrates the makespan values of the tasks submitted. The figure shows that MOGAMCC clearly outperforms MOSA and Min-Min in all the random cases. MOSA has better performance than the Min-Min algorithm. Although the difference is small, MOGAMCC achieved better performance against the MOSA in most cases. This is due to the parallel allocation of tasks to the available virtual machines. There is no starvation problem occurs in the MOGAMCC.

The utilization ratios of resources are plotted in Figure.11. The plot indicates that the resource utilization of both MOSA and MOGAMCC are almost same. The Min-Min algorithm has less utilization of resources when compared to other algorithms. This is due to the lack of rescheduling approach for the available VMs. But, in the MOGAMCC, the VM which completes the task execution will again reconsider as an available VM. Therefore, the resource utilization ratio for the proposed algorithm had better value against the Min-Min and MOSA.

Table 5 shows the comparison results of the MOSA, Min-Min and MOGAMCC tested with 300 tasks and 25 VMs. The proposed three objectives are implemented with the three algorithms and the results are shown. The MOGAMCC outperforms the remaining algorithms in all segments.

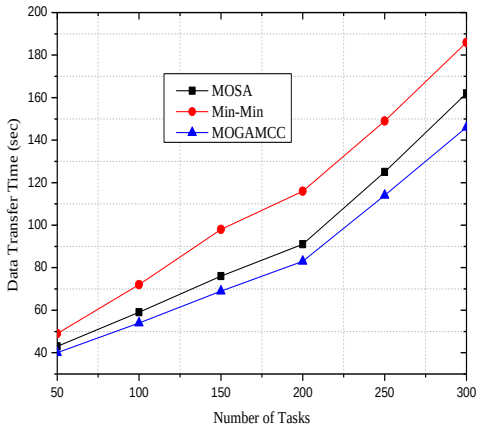

Figure.8 Data Transfer Time of the MOSA, Min-Min and MOGAMCC

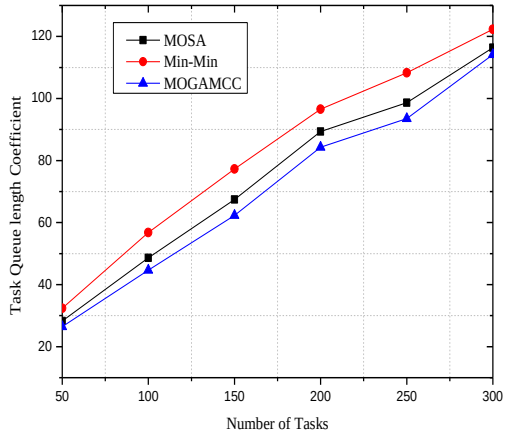

Figure.9 Task Queue length coefficient of the MOSA, Min-Min and MOGAMCC

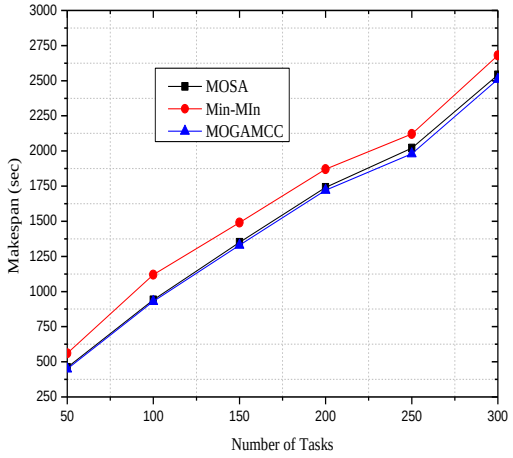

Figure.10 Makespan of the MOSA, Min-Min and MOGAMCC

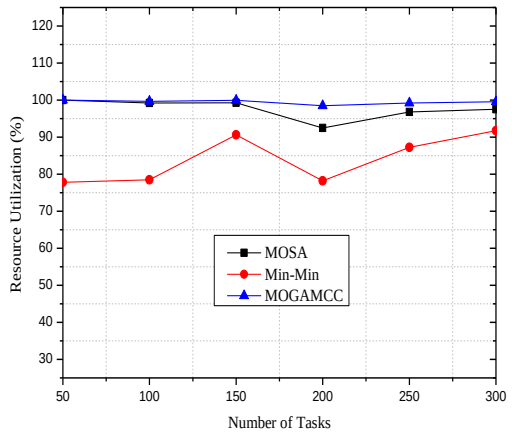

Figure.11 Resource utilization of MOSA, Min-Min and MOGAMCC

Table 5: Results Comparison

Optimization methods	**Data Transfer Time (sec)**	**Queue length**	**Makespan (sec)**	**Resource Utilization (%)**
MOSA	162	116.49	2540.70	97.52
Min-Min	186	122.36	2680.21	91.76
MOGAMCC	146	114.15	2510.32	99.59

7. Conclusion

Even though there are many existing resource scheduling approaches in the distributing environment, they have their own complexities in being directly applied to the MCC. This paper presents a multi objective genetic algorithm for resource scheduling in MCC. The proposed model introduced an encoding approach for finding the scheduling orders. The crossover and mutation process are applied for the generation of new population. The proposed model extended the functionality of cloudsim toolkit by implementing the multi objective genetic algorithm for achieving the better trade-off between the defined objectives. The extensive experiments are carried out on the MOGAMCC and the results are proved that this algorithm had the potentiality to minimize the objective functions.

In Future, the MOGAMCC is extended with fault tolerance mechanism which will be very useful at the time of VM failures.

References

[1] http://www.internetworldstats.com/asia/in.htm. Accessed on 24/04/2016.

[2] K. Kumar and Y.H. Lu, “Cloud computing for mobile users: Can offloading computation save energy?”, *Computer*, Vol. *43*, No. 4, pp.51-56, 2010.

[3] Z. Li, C. Wang, and R. Xu, “Computation offloading to save energy on handheld devices: a partition scheme”, In *Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems*, pp. 238-246, 2001.

[4] P. Rong and M. Pedram, “Extending the lifetime of a network of battery-powered mobile devices by remote processing: a markovian decision-based approach”, In *Proceedings of the 40th annual Design Automation Conference*, pp. 906-911, 2003, June. ACM.

[5] D. Nagaraju, and V. Saritha, "A Survey on Communicational Issues in Mobile Cloud

Computing." *Walailak Journal of Science and Technology (WJST)*, Vol. 14, No. 10, 2016.

[6] S. Ou, K. Yang, and J. Zhang, "An effective offloading middleware for pervasive services on mobile devices", *Pervasive and Mobile Computing*, Vol. 3, No. 4, pp.362-385, 2007.

[7] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive offloading inference for delivering applications in pervasive computing environments", In *Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on*, pp. 107-114, 2003, March. IEEE.

[8] J. Park, H. Kim, Y.S. Jeong, and E. Lee, "Two-phase grouping-based resource management for big data processing in mobile cloud computing", *International Journal of Communication Systems*, Vol. 27, No. 6, pp.839-851, 2014.

[9] B. Song, M. M. Hassan, and E. N. Huh, "A novel heuristic-based task selection and allocation framework in dynamic collaborative cloud service platform", In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pp. 360-367, 2010, November. IEEE.

[10] S. Tayal, "Tasks scheduling optimization for the cloud computing systems", *Ijaest-International Journal of Advanced Engineering Sciences and Technologies*, Vol. 1, No. 5, pp.111-115, 2011.

[11] J. F. Li, J. Peng, X. Cao and H. Y. Li, "A task scheduling algorithm based on improved ant colony optimization in cloud computing environment", *Energy Procedia*, *13*, pp.6833-6840, 2011.

[12] E. Juhnke, T. Dornemann, D. Bock, and B. Freisleben, "Multi-objective scheduling of BPEL workflows in geographically distributed clouds", In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 412-419, 2011, July.

[13] L. Guo, S. Zhao, S. Shen, and C. Jiang, "Task scheduling optimization in cloud computing based on heuristic algorithm", *Journal of Networks*, pp.547-553, 2012.

[14] W. Y. Shieh, and C. C. Pong, "Energy and transition-aware runtime task scheduling for multicore processors", *Journal of Parallel and Distributed Computing*, pp.1225-1238, 2013.

[15] X. Wang, Y. Wang, and Y. Cui, "A new multi-objective bi-level programming model for energy and locality aware multi-job scheduling in cloud computing", *Future Generation Computer Systems*, pp.91-101, 2014.

[16] N. Calheiros Rodrigo, R. Ranjan, A. Beloglazov, C. AF De Rose, and B. Rajkumar. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." *Software: Practice and Experience*, vol. 41, no. 1, pp. 23-50, 2011.

[17] M. R. Avinaash, G. R. Kumar, K.A. Bhargav, T.S. Prabhu, and D.I. Reddy, "Simulated annealing approach to solution of multi-objective optimal economic dispatch", In *Intelligent Systems and Control (ISCO), 2013 7th International Conference on*, pp. 127-132, 2013, January. IEEE.

[18] A. V. Karthick, E. Ramaraj, and R. G. Subramanian, "An efficient multi queue job scheduling for cloud computing", In *Computing and Communication Technologies (WCCCT), 2014 World Congress on*, pp. 164-166, 2014, February.

[19] D. Hadka, *MOEA Framework A Free Open Source Java Framework for Multi objective Optimization*, [Online], Available: http://www.moeaframework.org/.

[20] P. V. Krishna, S. Mishra, D. Naga Raju, V. Saritha and M.S. Obaidat, "Learning automata based decision making algorithm for task offloading in mobile cloud." *Computer, Information and Telecommunication Systems (CITS), 2016 International Conference on*. IEEE, 2016.

[21] D. Naga Raju, and V. Saritha. "Architecture for Fault Tolerance in Mobile Cloud Computing using Disease Resistance Approach." *International Journal of Communication Networks and Information Security (IJCNIS)*, Vol. 8, No.2, 2016.