PRIORITIZED GA-PSO ALGORITHM FOR EFFICIENT RESOURCE ALLOCATION IN FOG COMPUTING

Anu
Research Scholar, Computer Science and Engineering Department,
DCRUST, Murthal, Sonepat, Haryana, India
aujlan77anu@gmail.com

Anita Singhrova
Professor, Computer Science and Engineering Department,
DCRUST, Murthal, Sonepat, Haryana, India
nidhianita@gmail.com

Abstract - Everyone is connected to internet these days and internet plays a vital role in daily lives of humans. Ubiquitous devices like smart phones makes internet accessible anywhere. Key issue into day's computing environment is how to efficiently address the demands of billions of internet users. In computing environment, extension of cloud computing has been introduced, named as Fog computing. Fog computing provides services in the proximity of end user with very low latency. It is an intermediate layer between cloud and IoT devices. Fog layer's main objective is to reduce the delay and improve the response time for user's request. To use the Fog Computing effectively, the resources should be allocated in an efficient way, while catering to the user requests. In this paper, a hybrid Prioritized Genetic Particle Swarm Optimization (P-GA-PSO) algorithm has been proposed for efficient resource allocation in fog computing. This proposed algorithm allocates tasks to the resources efficiently, consequently reducing delay, waiting time and energy consumption by 8.73%, 22.65% and 17.81% respectively as well as improving resource utilization by 0.54% in comparison to GA. Similarly, the proposed algorithm when compared with Round Robin algorithm showed reduced delay, waiting time and energy consumption by 3.90%, 21.99% and 1.68% respectively as well as improving resource utilization by 12.51%. Further, the quantitative analysis showed that the proposed algorithm performs better than GA and round-robin algorithms and moves towards optimal solutions faster than these algorithms.

Keywords: Fog; Edge Devices; Resource Allocation; GA; PSO; Prioritization.

1. Introduction

Fog computing [Hu *et al.,* (2017)] is a non-trivial extension of cloud computing paradigm which provides computing, storage and networking services at the edge of the network. Fog computing was introduced to deal with ever increasing demands of IoT devices and to address the challenges faced by traditional cloud computing framework like unreliable latency, lack of mobility etc.[Mahmud *et al.,* (2018)].Fog works as a virtualized intermediate layer between end devices and traditional cloud computing data centers. It is a model which decentralized computational resources towards edge of the network to improve quality of service: minimize delay and improve execution time of application requests.

1.1 *Resource allocation in fog computing*

Fog computing comes with its new set of challenges and problems in the computing world. To take advantage of Fog computing it becomes necessary to develop an efficient resource allocation and management [Ghobaei-Arani *et al.,* (2019)] techniques.

Fog environment encompasses limited resources, heterogeneous networks and dynamic fog devices. Resources used by Fog and Cloud are similar (computation, storage and network) but Fog layer have resources with limited storage and computing power, it is cumbersome to run multiple VMs simultaneously Thus resource allocation in fog computing becomes much more complex. Proper management [Mohan and Raj, (2012)] of resources in such situation has become one of the main areas of research in fog computing.

Resource allocation is performed based on several parameters, so as to increase overall performance of fog servers. Each task in fog computing is being assigned to one of the available fog servers. In turn, fog server processes the requested task and send response to the user. A good resource allocation algorithm minimizes the delay and response time and maximize the resource utilization. Geographically distributed fog nodes should be allocated in efficient and fair manner for competing IoT services according to the requirement of services.

The related work is included in section 2, followed by proposed Prioritized-GA-PSO algorithm in section 3. Section 4 presents the results. The paper is finally concluded in section 5.

2. Related Work

Research in the field of fog computing is in nascent stage. Therefore, not much of the research has been done on resource management in fog computing.

In 2017, Bittencourt *et al*. introduced scheduling policies which considered user mobility and edge computing capacity in context of fog computing infrastructure. They also discussed the problem caused in resource allocation due to hierarchical infrastructure along with different scheduling policies [Bittencourt *et al.,* (2017)]. An algorithm was proposed by Haruna *et al*. for seamless handover and some scheduling policies to tackle the mobility of edge devices were also suggested by them [Haruna *et al.,* (2017)]. A knapsack and knapSOS based scheduling was presented by Dadmehr *et al.* which was optimized using symbiotic organisms search [Rahbari and Nickray, (2017)]. Khurram *et. al*. enhanced AODV with link prediction model with GA to improve route maintenance phase. AODVLGA improved end to end delay, improvised packet delivery ratio and lowered the routing overhead [Khurram and Dr., (2020)].

Sun Yan *et al*. proposed a fog computing structure and presented a crowd-funding algorithm, so that spare resources in the network can be pooled together [Sun and Zhang, (2017)]. They proposed an incentive mechanism to encourage more resource owner to contribute their spare resources so that increased number of tasks could be executed successfully. Bitam *et al.* presented a bio inspired optimization approach. They implemented Bees Life Algorithm (BLA) to distribute the tasks over all fog nodes and to address the challenges of job scheduling in fog environment [Bitam *et al.*, (2018)]. Ni Lina *et al*. proposed priced timed petri nets (PTPNs) based resource allocation strategy for fog computing, through which user could choose the desired resources autonomously from a group of pre-allocated resources [Ni *et al.*, (2017)]. A workload allocation methodology in cloud-fog environment for minimizing the energy consumption and transmission delay was provided by Deng *et al.* They decomposed the key problem into subproblems which could be solved within corresponding subsystems respectively [Deng *et al.*, (2016)]. Kochar *et al*. presented two scheduling algorithms in fog environment Distributed Earliest Deadline First for fog and Profit-aware Distributed EDF for fog to maximize resource utilization of the end devices under a service provider [Kochar and Sarkar, (2016)]. A methodology was proposed by Aazam *et al*. for resource estimation and management. On the basis of several factors such as relinquish probability of customer, service type, service price and variance of relinquish probability they formulated resource management and allocation in advance [Aazam and Eui-Nam Huh, (2015)]. Their methodology helped in determining the precise number of resources required and avoiding resource wastage. Agarwal *et al.* presented a survey on virtualization techniques and proposed architecture for elastic resource allocation [Agarwal *et al.*, (2015)].

Unlike these research studies, this paper presents an efficient algorithm for resource allocation in fog computing which combines the strength of priority algorithm, genetic algorithm and particle swarm optimization algorithm. The effectiveness and efficiency of proposed algorithm is established by comparing it with GA and Round-Robin algorithm, for resource allocation in fog environment.

3. Proposed Algorithm

The proposed algorithm is combination of priority, GA and PSO algorithm, namely P-GA-PSO algorithm. There are some limitations of GA and PSO: (i) Computation time of GA is very high, (ii) execution time of PSO is better than GA and (iii) solution provided by PSO is better than GA but PSO algorithm has tendency to get stuck in the local optimal solution and due to premature convergence, it cannot be scaled to accommodate large number of requests. Hence, in order to get best performance a hybrid P-GA-PSO is proposed. The proposed algorithm executes faster and does not get stuck in local optimum solution. Also, GA mutation operator provide more improved and accurate solution. The main objective is to reduce delay, waiting time and energy consumption along with maximized resource utilization.

The P-GA-PSO algorithm starts with selecting best available VMs depending upon their priority. Priority is assigned to VMs depending on their status i.e. VM is free or busy and total count of tasks allocated to that VM. The priority assigned to i^{th} VM is given in Eq. (1):

$$VM - Priority_i = f\left(VM_i\, is \frac{free}{busy}, already\ allocated\ task\ to\ VM_i\right) \quad (1)$$

Then random population is generated which consist number of chromosomes. Fitness value is calculated for each chromosome depending upon mapping of tasks over selected VMs. These chromosomes are solutions to the resource allocation problem, Solution is basically distribution of tasks over available VMs. Thereafter, GA algorithm is applied over initialized population and GA is repeated *n* times so that best chromosomes will be selected and passed as particles to PSO algorithms. PSO algorithm is applied over these particles till max epoch. At each iteration every particle moves in direction to achieve best optimal solution. Particles with maximum fitness values are selected to get the optimal solution for resource allocation problem. The details of P-GA-PSO algorithm are presented in Figure 1.

```
Input: List of Available VMs
Output: Best and Optimized Position of VMs

1. Create VM_list                          //list of VMs at fog layer
2. if (VM state == available && (current_allocation_size < min_allocation_size))
                add VM_id in priority queue
3. VMs added in priority queue are selected to generate random population
4. Initialize a random population   //population contains several solutions, solution is allocation
                                    //of tasks over VMs (these solutions are called chromosomes)
5. compute fitness of every chromosome
6. for i=0 to n                            // n is the maximum number iterations
   {
        Call selection operator on population to select fittest chromosomes
        Perform crossover on selected chromosomes to generate offspring
        Perform mutation on offsprings to generate new_chromosome
   }
7. set new_chromosome as particle for PSO
8. Initialize velocity and position of particle randomly
9. Compute Pbest and Gbest values
10. For all particles
    {
    Update velocity using Eq. (1)
    Update position using Eq. (2)
     }
11.  If iteration < max
        repeat from step8.
```

Fig. 1. P-GA-PSO Algorithm

3.1 *Priority based selection of VM*

Initially a VM list of available VMs at fog layer is created, then these VMs are selected based upon a priority algorithm.VM state will be either busy or available.VM id of those VMs is being returned in priority queue which are currently in available state and executing minimum number of tasks. The algorithm is shown in Figure 2.

```
Input: List of Available VMs
Output: VM_id of Prioritized VMs
for each available VM
{
   if (VM state == available && (current_allocation_size < min_allocation_size))
   {
            return VM_id
   }
}
```

Fig. 2. Algorithm for selection of VM on Priority basis

3.2 *Genetic algorithm*

The concept of GA is based upon theory of evolution which explain origin of species, according to which strong and fittest species have opportunity to participate in reproduction to pass their genes in coming generations while unfit and weak species will be discarded or extinct by natural selection [Konak *et al*., (2006)].

Genetic algorithms are efficient to find the global optimum solution where sample space is large and complex. These are not only used to find the optimum solution for single objective problems but also beneficial to find the optimum solution for multi-objective problems [Deb *et al.,* (2002)].This is the reason why we prefer them in resource scheduling in fog computing environment. The main steps of GA are as follow:

3.2.1 Generate random population

The GA algorithm starts with generating random population, population represents several solutions. Solutions are mapping of tasks over VMs available at fog servers. These solutions are called as chromosomes. Genes of chromosome represents individual VMs and length of chromosome is equal to number of tasks to be executed. GA primarily works on first randomly generated population of prioritized VMs.

```
Input: Prioritized VMs
Output: Random population of prioritized VM Group
For i=0 to n        // n is the size of population
{
        Add random prioritized group of VMs to initial population
    }
```

Fig. 3. Algorithm for generation of initial population

3.2.2 Selection operator

Selection is an approach to keep fittest chromosomes for next generation and discard remaining chromosomes from the population. It is a simple ranking approach in which survival of a chromosome for next generation depends upon its fitness value. The fitness value is calculated by counting total number of tasks allocated in a VM group given by Eq. (2). So, during selection operation fittest group of VMs is being selected from population of prioritized VMs group.

$$Total\ Fitness = \sum_{i=1}^{n} Total\ tasks\ allocated \quad (2)$$

where n= Total number of VMs in VM Group.

```
Input: Random Population of Prioritized VM Group
Output: Fittest Group of VMs
Set the selection size =n
For i=0 to selection size
        {
          Return the fitness value of each VM Group
          Select the VMs Group with maximum fitness value
          }
```

Fig. 4. Algorithm for selection of fittest VM Group

3.2.3 Crossover

In crossover two fittest groups of VMs are allowed to combine together and reproduce so that they can exchange their genes with each other to reproduce new group of VMs. Crossover is an important operator of genetic algorithm to produce new population with better fitness value. The new chromosome generated after crossover are called as offspring. Here, single-point crossover is being used where middle crossover point is selected and tails of two parents are being swapped to get new offspring.

```
Input: Two Fittest Group of VMs                                    // Single Point Crossover
Output: New Group of VMs i.e. offsprings
For i=0 to n/ 2                                                    //n is the population size
{
      Chromosomes i.e. two fittest groups of VMs.                  // offsprings are generated with
       For j=0 to length of chromosome                             // recombination of two parent
        {
           Offspring1= parent1.substring(0, length/2) + parent2.substring(length/2, length)
            Offspring2= parent2.substring(0, length/2) + parent1.substring(length/2, length)
          }
 }
```

Fig. 5. Algorithm for Crossover to Reproduce New Group of VMs

3.2.4 Mutation

Mutation introduces random changes in characteristics of newly produced chromosomes. Mutation operator is applied on genotypes at gene levels with a p_mutation probability, which is very small in rate. One or more genes i.e. VMs are being selected for mutation and to generate modified group of VMs.

```
Input: New Group of VMs
Output: Modified Group of VMs after Mutation
If(Mutate==True)
{
Set num =length of offspring
index = (int)(Math.random()*num)                          //select a random position in offspring
newBit = flip(offspring.substring(index, index+1))              //flip method convert 0 into 1
newBitString = offspring.substring(0, index) + newBit +          //  and 1 into 0
offspring. Substring(index+1 , offspring.length())          // New chromosome generated after
}                                                                  //mutation
```

Fig. 6. Algorithm for Mutation

3.3 *PSO*

Particle swarm optimization is a population based meta heuristic search algorithm that is based on swarm intelligence and was developed by kennedy and Eberhart in 1995 [Kumar and Sharma, (2019)]. It is inspired by the social behavior in animals and human beings for example bird flocking. PSO is very much similar to genetic algorithm (GA), but it has some advantages over GA i.e. its implementation is easy, has less parameters to adjust and have higher efficiency in problem solving.

In most cases PSO finds optimal solutions to various problems using Branch and Bound algorithm. Thus, PSO is used to solve resource management problem in fog computing. Main objective of PSO is to converge to the best value of fitness function after every iteration.

In PSO every individual of swarm represents a feasible solution. This individual is known as particle [Kumar and Raza, (2015)]. Population of PSO is the total number of particles that is generated randomly at the initial phase of algorithm. Every particle can move in Multi-Dimensional search space which is represented by two parameters namely Position and Velocity. During flights in multi-dimensional search space particles adjust their velocities according to some rules and change their position by moving towards better position [Kennedy' and Eberhart, (1995)]. While changing its position in each iteration k, each particle stores two values: local best (best position of particle itself) and global best (the best position of particle among whole population).

3.3.1 Update velocity

Pbest and Gbest are used for velocity updation of particles and the change in velocity in every iteration for creating a new solution is defined in Eq. (3).

$$V_i^d(\mathrm{t}+1) = V_i^d(\mathrm{t}) + \mathrm{c1r1}(\mathrm{Pbest}_i^d(\mathrm{t}) - X_i^d) + \mathrm{c2r2}(\mathrm{Gbest}_i^d(\mathrm{t}) - X_i^d) \quad (3)$$

Where:

c1, c2 – Acceleration Coefficients

r1, r2 - uniformly distributed numbers in interval [0,1]

Input: Modified Group of VMS after Mutation
Output: Updated Fitness values of VMs
For i=1 to pcount //pcount is number of particles
{
$V_i^d(t+1) = V_i^d(t) + c1r1(Pbest_i^d(t) - X_i^d) + c2r2(Gbest_i^d(t) - X_i^d)$
}

Fig. 7. Algorithm for velocity updation

Velocity of particles i.e., individual VM is being changed to attain better fitness value and consequently generate new updated group of VMs.

3.3.2 Update position

After updating velocity Eq. (4) is used to update the position of VMs depending upon their velocity.

$$X_i^d(t+1) = X_I^d(t) + V_i^d(t+1) \quad (4)$$

Input: Updated Fitness values of VMs
Output: Best and Optimized Position of VMs in Group i.e. Optimal Solution
For i=1 to pcount
{
$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1)$
}

Fig. 8. Algorithm for position updation

4. Simulation

The P-GA-PSO algorithm is implemented and evaluated through iFogSim simulator. This section provides a brief detail of simulator used and simulation parameters.

4.1 Simulator and simulator settings

To evaluate the performance of proposed algorithm, the proposed Prioritized GA-PSO algorithm is implemented using the iFogSim which extends the existing CloudSim simulator.

The set of simulation parameters which are used to perform simulation and record the performance metrices' values is presented in Table 1. The assumed values of P-GA-PSO parameters are presented in Table 2.

Table 1. Simulation Parameters

Parameter	Values
Min number of edge devices	100
Max number of edge devices	400
Number of tasks in application	100-800
MIPS	2000
RAM	4000 MB
Bandwidth	300 mbps
VM Policy	Space shared

Table2. Prioritized GA-PSO Algorithm Parameters

Parameter	Value
Number of chromosomes	10
Mutation Rate	0.5
Crossover	Single Point
Fitness threshold	5
Swarm Size	10
Max Epochs	500

4.2 Simulation results

This section represents the simulation results and their analysis. In context of resource scheduling the proposed P-GA-PSO algorithm is compared with two existing algorithms, traditional and intelligent algorithm. In traditional algorithms Round- Robin algorithm is being selected because it uses fair selection policy for scheduling which results in decreased execution time and waiting time. GA is being selected as intelligent algorithm for comparison because it is general algorithm and its flexible and scalable nature make it applicable in broad class of optimization problems. The comparison and analysis are done using four parameters namely, a) Average CPU Utilization, b) Execution Delay c) Waiting Time d) Energy Consumption.

4.2.1 Average CPU utilization

The main aim of this research is to maximize the CPU utilization as much as possible. The impact of increase in number of edge devices on CPU Utilization is shown in Figure 9. The mapping of tasks over best suitable VMs in proposed algorithm prevents CPUs to compromise their performance without going underutilized as well over allocation to minimize the context switching. So, Average CPU utilization of P-GA-PSO is better than both Round Robin and GA algorithms for large number of edge devices.

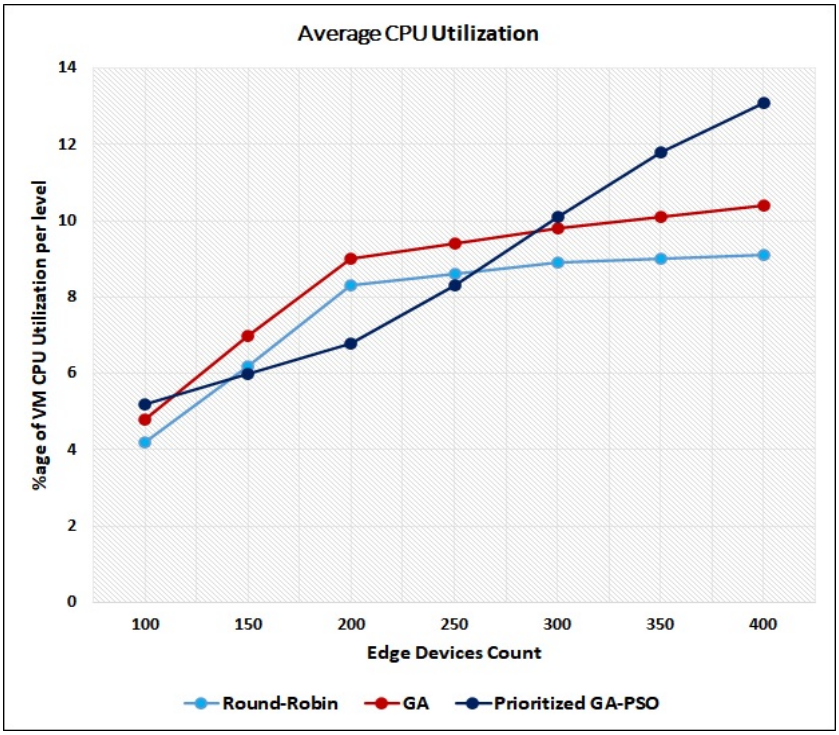

Fig. 9. Average CPU Utilization

4.2.2 Execution delay

One of the key challenges in fog computing is to minimize the delay. It is required as some applications like augmented reality, event processing etc. are latency sensitive. The comparison of execution delay of P-GA-PSO, GA and Round-Robin is shown in Figure 10. It has been observed that execution delay of P-GA-PSO is lowest with the increased number of edge devices because proposed algorithm always chose best VM, so that selected tasks is executed with minimum execution delay.

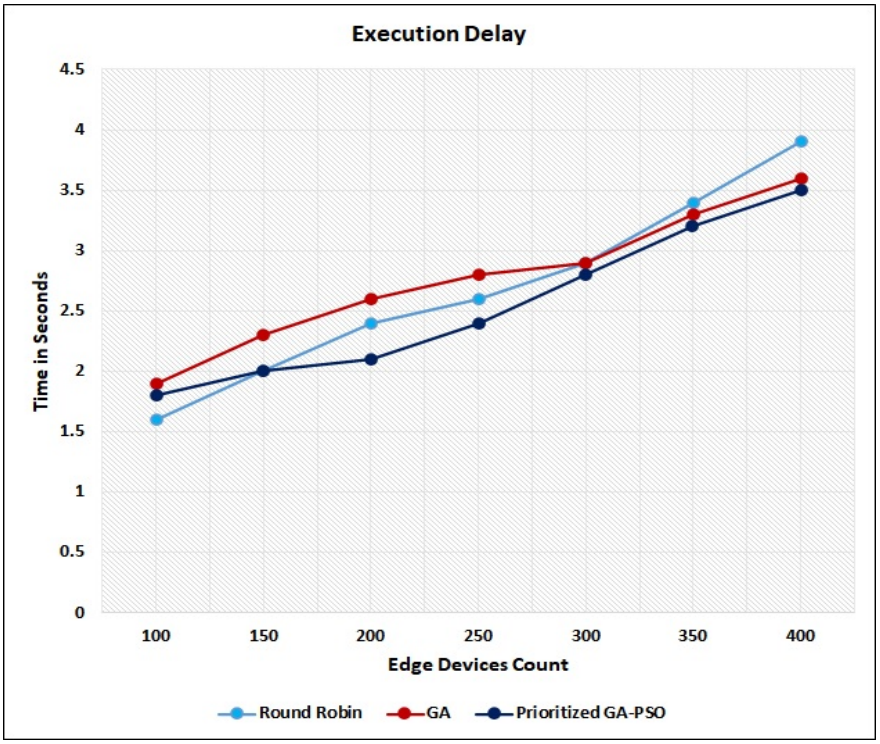

Fig. 10. Execution Delay

4.2.3 Average waiting time

Waiting time is the time spent by tasks in waiting queue for resource allocation. One of the objectives of the proposed algorithm is to minimize the waiting time of tasks to be executed. The comparison of average waiting time of P-GA-PSO, GA and Round-Robin with varying number of nodes is shown in Figure 11. Selection of appropriate VM for task execution minimizes the average waiting time in P-GA-PSO algorithm w.r.t. comparison with GA and Round-Robin algorithm.

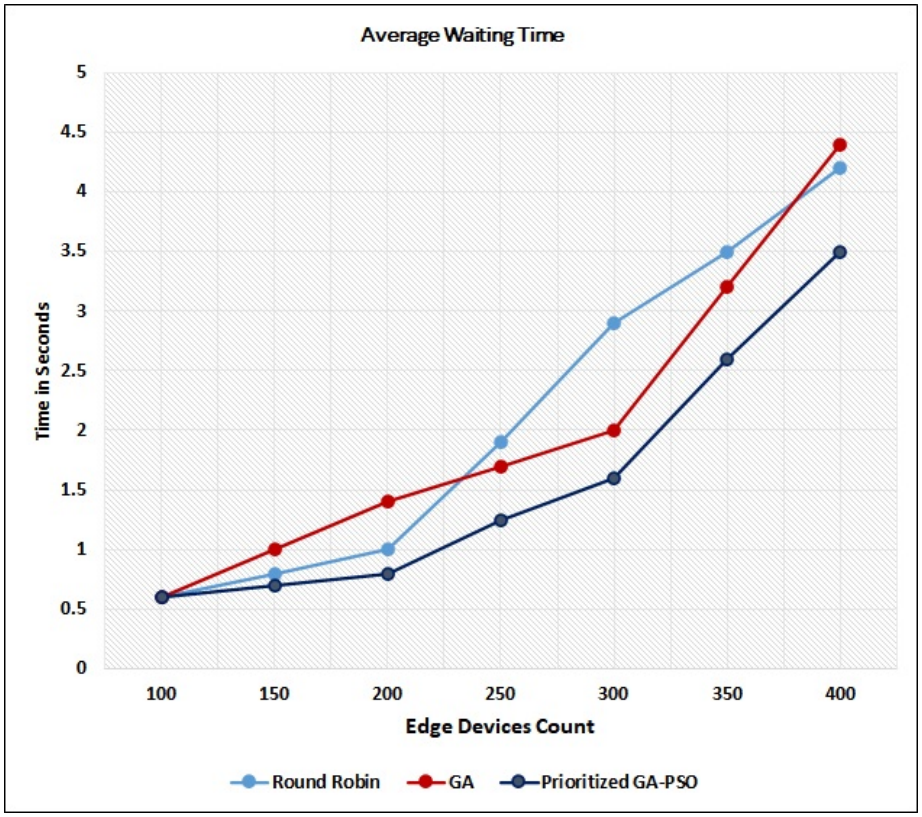

Fig. 11. Average Waiting Time

4.2.4 Energy consumption

The variations in energy consumption of different algorithms are depicted in Figure 12. It has been observed that for lower number of edge devices GA consumes maximum amount of energy. In spite of a steep downward plunge when edge devices scaled from 100 to 150 GA still consumes more energy than P-GA-PSO and Round-Robin. The P-GA-PSO and Round-Robin which consumes almost same amount of energy. But as the number of edge devices increases energy consumption in Round-Robin start increasing and in P-GA-PSO energy consumption is minimized due to appropriate mutation rate and PSO algorithm converge the solution towards optimal one.

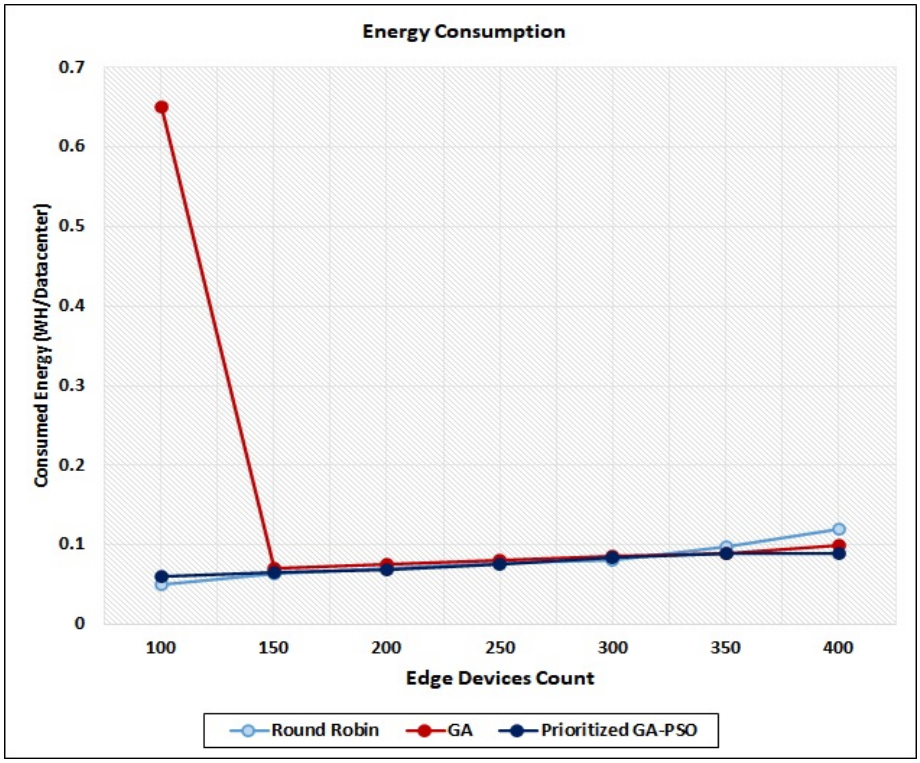

Fig. 12. Energy Consumption

Table 3. Simulator Results of Prioritized GA-PSO, GA and Round-Robin

Edge Device Count	%age VM CPU Utilization			Execution Delay (in Seconds)			Waiting Time (in seconds)			%age of Energy Consumption (in Watt per Hour)		
	RR	GA	P-GA-PSO	RR	GA	P-GA-PSO	RR	GA	P-GA-PSO	RR	GA	P-GA-PSO
100	4.2	4.8	5.2	1.6	1.9	1.8	0.6	0.6	0.6	0.050	0.650	0.060
150	6.2	7.0	6.0	2.0	2.3	2.0	0.8	1.0	0.7	0.063	0.070	0.065
200	8.3	9.0	6.8	2.4	2.6	2.1	1.0	1.4	0.8	0.070	0.075	0.068
250	8.6	9.4	8.3	2.6	2.8	2.4	1.9	1.7	1.25	0.077	0.080	0.075
300	8.9	9.8	10.1	2.9	2.9	2.8	2.9	2.0	1.6	0.081	0.085	0.084
350	9.0	10.1	11.8	3.4	3.3	3.2	3.5	3.2	2.6	0.098	0.090	0.090
400	9.1	10.4	13.1	3.9	3.6	3.5	4.2	4.4	3.5	0.120	0.100	0.090

4.3 Analysis of P-GA-PSO

This section highlights the relevance of proposed P-GA-PSO to improve average CPU utilization, execution delay, average waiting time and energy consumption under varying number of edge devices. Round Robin algorithm is totally dependent upon the size of time quantum. If we choose large time quantum it shows results similar to FCFS & very small-time quantum causes high context switches and minimized CPU utilization. GA executes slowly and has higher waiting time and delay than our proposed algorithm. The quantitative comparison of P-GA-PSO w.r.t. Round-Robin and GA algorithms is summarized in Table 4.

Table 4. Comparison of Prioritized GA-PSO with GA and Round-Robin

Parameter	with GA	with Round-Robin	Impact
Average CPU Utilization	+0.54%	+12.51%	Improvement
Execution Delay	-8.73%	-3.90%	Improvement
Average Waiting Time	-22.65%	-21.99%	Improvement
Energy Consumption	-17.81%	-1.68%	Improvement

From the result analysis, it is clear that the proposed algorithm allocates the resources to tasks in fog environment in efficient and effective manner, so as to minimize the delay, waiting time and energy consumption and maximize the CPU utilization. Hence, the proposed algorithm achieves the objective of allocating best and suitable resources to the client requests immediately in optimal and efficient way.

5. Conclusion

Fog computing is an emerging paradigm that offers storage and computation facility resources at the proximity of end devices. With the increasing prevalence of fog computing, the resource allocation to end user requests has become a relevant research issue. It aims for achieving the minimized resource wastage, execution delay, waiting time as well to save energy. Thus, a hybrid P-GA-PSO resource allocation algorithm has been proposed. The quantitative results of proposed algorithm were compared with Round-Robin and GA algorithms. The task allocation done using proposed algorithm showed reduced delay, waiting time and energy consumption by 8.73%, 22.65% and 17.81% respectively as well as improved resource utilization by 0.54% in comparison to GA. Similarly, the proposed algorithm when compared with Round Robin algorithm showed reduced delay, waiting time and energy consumption by 3.90%, 21.99% and 1.68% respectively as well as improving resource utilization by 12.51%. Further, a quantitative analysis presented in this work showed improved performance of proposed P-GA-PSO.

References

[1] Aazam M. and Huh Eui-Nam, (2015): Dynamic resource provisioning through Fog micro datacenter, IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), St. Louis, MO, pp. 105–110, doi: 10.1109/PERCOMW.2015.7134002.

[2] Agarwal S., Yadav S., and Yadav A. K, (2015): An architecture for elastic resource allocation in Fog Computing, vol. 6, no. 2, pp. 7.

[3] Bitam S., Zeadally S., and Mellouk A., (2018): Fog computing job scheduling optimization based on bees swarm, Enterp. Inf. Syst., vol. 12, no. 4, pp. 373–397, doi: 10.1080/17517575.2017.1304579.

[4] Bittencourt L. F., et. al. (2017): Mobility-Aware Application Scheduling in Fog Computing, IEEE Cloud Comput., vol. 4, no. 2, pp. 26–35, doi: 10.1109/MCC.2017.27.

[5] Deb K.,et. al., (2002): A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 182–197, doi: 10.1109/4235.996017.

[6] Deng R., et. al., (2016): Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption, IEEE Internet Things J., vol. 3, no. 6, pp. 1171–1181, doi: 10.1109/JIOT.2016.2565516.

[7] Ghobaei-Arani M., Souri A., and Rahmanian A. A., (2019): Resource Management Approaches in Fog Computing: A Comprehensive Review, J. Grid Computing, vol. 18., pp.1-42 doi: 10.1007/s10723-019-09491-1.

[8] Hu P., et.al., (2017): Survey on fog computing: architecture, key technologies, applications and open issues, J. Netw. Comput. Appl., vol. 98, pp. 27–42, doi: 10.1016/j.jnca.2017.09.002.

[9] Kennedy J. and Eberhart R., (1995): Particle swarm optimization, Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, vol.4, pp. 1942-1948, doi: 10.1109/ICNN.1995.488968.

[10] Khurram, M., Biradar Dr., A., (2020): A link based genetic algorithm approach in optimizing routing in wireless adhoc network, Indian J. Comput. Sci. Eng. 11, pp. 488–496. https://doi.org/10.21817/indjcse/2020/v11i5/201105112

[11] Kochar V. and Sarkar A., (2016): Real time resource allocation on a dynamic two-level symbiotic fog architecture, Sixth International Symposium on Embedded Computing and System Design (ISED), Patna, India, pp. 49–55, doi: 10.1109/ISED.2016.7977053.

[12] Konak A., Coit D. W., and Smith A. E., (2006): Multi-objective optimization using genetic algorithms: A tutorial, Reliab. Eng. Syst. Saf., vol. 91, no. 9, pp. 992–1007, doi: 10.1016/j.ress.2005.11.018.

[13] Kumar D. and Raza Z., (2015), A PSO Based VM Resource Scheduling Model for Cloud Computing, IEEE International Conference on Computational Intelligence & Communication Technology, Ghaziabad, India, pp. 213–219, doi: 10.1109/CICT.2015.35.

[14] Kumar M. and. Sharma S. C, (2019): PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing, Neural Comput. Appl., vol.32, pp. 12103-12126, doi: 10.1007/s00521-019-04266-x.

[15] Mahmud R., Kotagiri R., and Buyya R., (2018): Fog Computing: A Taxonomy, Survey and Future Directions, in Internet of Everything, B. Di Martino, K.-C. Li, L. T. Yang, and A. Esposito, Eds. Singapore: Springer Singapore, pp. 103–130.

[16] Mohan N. R. R. and Raj E. B., (2012): Resource Allocation Techniques in Cloud Computing -- Research Challenges for Applications , Fourth International Conference on Computational Intelligence and Communication Networks, Mathura, Uttar Pradesh, India, pp. 556–560, doi: 10.1109/CICN.2012.177.

[17] Name H. A. M., Oladipo F. O., and Ariwa E., (2017): User mobility and resource scheduling and management in fog computing to support IoT devices, Seventh International Conference on Innovative Computing Technology (INTECH), Luton, pp. 191–196, doi: 10.1109/INTECH.2017.8102447.

[18] Ni L., et. al, (2017): Resource Allocation Strategy in Fog Computing Based on Priced Timed Petri Nets, IEEE Internet Things J., vol. 4, no. 5, pp. 1216–1228, doi: 10.1109/JIOT.2017.2709814.

[19] Rahbari D. and Nickray M., (2017): Scheduling of fog networks with optimized knapsack by symbiotic organisms search, 21st Conference of Open Innovations Association (FRUCT), Helsinki, pp. 278–283, doi: 10.23919/FRUCT.2017.8250193.

[20] Sun Y. and Zhang N., (2017): A resource-sharing model based on a repeated game in fog computing, Saudi J. Biol. Sci., vol. 24, no. 3, pp. 687–694, doi: 10.1016/j.sjbs.2017.01.043.