LATENCY CONTROL USING FOG-BASED SPIDER WITH MULTI-STAGE SEARCH (FSMS) ALGORITHM IN MOBILE COMPUTING

J. RATHIKA[1], Dr. M. SORANAMAGESWARI[2]

([1]Assistant Professor, Department of Computing, Coimbatore Institute of Technology, Coimbatore, India. Email: rathikaj300@mail.com)

([2]Assistant Professor, Department of Information Technology, Government Arts College, Coimbatore, India)

ABSTRACT

Users may access computer resources from any location at any time thanks to the network-based computing paradigm known as cloud computing. There has been an enormous quantity of data created recently as the Internet-of-Things (IoT) sector leveraging the cloud has expanded, and the demand for services needing low latency is rising. Fog (cloud) computing is a novel architecture that has been developed to address these issues. Fog computing using Fog based Spider with Multistage Search (FSMS) algorithm can handle data processing on a network device nearby the user. Fog based Spider (FS) enhances the services offered in terms of the key techniques for service configuration such as symmetry and asymmetry by using webs in spider. Multistage Search (MS) reduces the number of integrated web services, which improves the spider web service selection and setup operations, using the mutual FS and MS. By combining these two methods, far less network bandwidth is used while yet delivering nearly instantaneous responses. The best network devices to employ to install the fog-based spider with a multi-search strategy haven't received much investigation, though. We advise and recommend the deployment method in this study work for a fog server in this computing environment. This is used to lessen the data movement path and a technique to make full use of the computing resources of a fog device concentrated on single network device with the combined evaluation of minimum distance and angle. The suggested technique can maximise the use of the mobile computing resource allocations of the fog or cloud devices, according to experimental results, which also demonstrate that it can minimise the data transfer distance.

Keywords: Cloud computing, resource allocations, Internet-of-Things (IoT), Fog based Spider with Multistage Search (FSMS), Multistage Search (MS), DOG-FPA

2. INTRODUCTION

Users may access computer resources from any location at any time thanks to the network-based computing paradigm known as cloud computing. A distributed platform named for cloud computing can allow users to access servers and cloud massive data centers from different locations. Different paradigms (public, private, mixture, and community) are used, and various services (software, platforms, infrastructure, and networks) are provided globally [1]. Due to the complicated and crowded nature of the resources across the distributed cloud servers, resource distribution on cloud computing platforms can occasionally become a major source of aggravation or even disaster. As a result, the workload allocation is not always smooth since the complexity of cloud latency from the many participating compute nodes via the network is increased due to congestion over the core-cloud network [2].

By transforming the network's processing capability from the core to the edge into a dispersed decentralised computing area, Cisco has increased the potential of cloud computing. Fog computing is a new computing paradigm. In the atmosphere of IoT (Internet of Things), billions of edge linked devices create fog computing like the perfect foundation for applications and services. The equivalent computational functions (networking, storage, computing) are provided as in the cloud, but because computing is done more nearer to the user's edge parts or components, it enables more intent and closeness [3]. Fog is positioned near and below the cloud, acting as a layer between the major data centres and edge devices [4].The hierarchical architecture overview of fog computing is shown in Fig. 1.

The fact that it is situated close to grounded edge devices makes them the most effective and beautiful platform for real-time apps and services. Instead of transferring data to bulk cloud data centres, fog-nodes enable data processing at the network's edge [5]. Comparing the proposed FS technique to the conventional cloud computing, which is a requirement for the applications and services operating in an IoT domain, shows that it improves proximity searching, lowers latency complexity, and provides faster reaction times. To lessen network bottlenecks and congestions caused by data communication and transformation, workloads are temporarily offloaded to fog nano data centres.

Cloud customers are growing impatient as a result of delays in web application content loading over the internet, which is typically brought on by complex latency when accessing cloud datacenters located far away from the cloud users. Utilizing the apps and services over the cloud-centric network is quickly turning into a catastrophe. The numerous layers of the cloud's workload distribution also add to the latency. Time-sensitive (IoT) Internet of Things applications and services typically run on multiple virtual machines (VMs) in a cloud platform and are highly complex when interacting. They encounter challenges when attempting to consolidate the several apps with heterogeneous workloads.

Cloud computing services are brought using fog computing to the network edge, where processing, storage, and communication, closest with the edge components of the end user. As a result, it makes the most network bandwidth in available, enhances mobility, and reduces latency. It searches nearby points, [6] (i.e., the closest and nearest points) as per the Markov chain. The fog node reduces reaction times and service delays by sitting among cloud and the IoT edge devices. In order to avoid congestion that causes the greatest service delays or else latency with complications, data transferred from a user's edge device to a fog node is offloaded to the nearest accessible fog nodes for the least amount of delay or waiting time, or else to the data centers in cloud. Both wired and wireless LANs are used by edge devices in IoT to communicate requests to the neighbourhood points. Response time will be minimal as long as the request falls within the serving domain range (RL). Users may encounter the greatest response delays once their data crosses the barrier (CL). To reduce Round-Trip Time (RTT), fog nodes are positioned between the vital cloud and a IoT-edge platform [7].

The aforementioned studies only take into account the use of cloud resources after edge resources have been used up by dumping workloads. This might result in too many underutilised cloud resources. Our study thus intends to provide a combination approach of service caching, computational resource allocation, and communication channel assignment in order to maximise user pleasure and resource efficiency. In order to promote more collaboration across edges and clouds, we also aim to capitalise on

their heterogeneity. These apps are also a part of cloud computing technologies that primarily allocate resources through mobile devices.

3. RELATED WORKS

In a previous paper [8], the authors suggested an analytical approach for reducing service delays in an Internet of Things context. The QoS for the time-sensitive IoT applications was automatically enhanced by the reductions in delays. The authors of a different study [9] suggested the cutting-edge swarm Social Spider Cloud Web Algorithm, an intelligence-based metaheuristic, which would help to solve the problems with workload distribution in the centralised cloud computing platform. To locate as many prey sites as possible within the closest local geographic areas, the foraging behavioural mechanism was applied. In a different paper [10], the authors suggested and developed a convex optimised distributed method Based on the proximate algorithm, video streaming using fog computing will improve shared resource allocation and reduce carbon footprints. Their method was developed to deconstruct difficult issues into more manageable issues.

In other [11], the authors looked into the characteristics of fog systems' task completion latencies at six heterogeneous sites using benchmarking data. Their findings in the conducted work reflected the ideal latencies. They had a strategy in place. The authentications of their recommended algorithms, however, were referred to be their cutting-edge work. In a different study [12], the authors developed a novel consolidation methodology based on profiling, with the primary goal of lowering the number of physical computers in order to optimise tail latency. The proposed method can significantly minimise the tail latency when compared to the conventional consolidation method, according to experimental findings. The authors discussed the smart gateways for the fog computing platform in another article [13]. We adjusted a number of delay characteristics, including synchronisation, uploading, and network jitters. It has been demonstrated that smart gateway-based communication may ease the strain and improve cloud resource provisioning, leading to effective use.

The authors suggested deploying fog servers in order to slow down data transmission in a fog computing environment. They employed a vector bin packing algorithm as a component of a method to utilise all of the devices' resources. The cost-based task scheduling technique was developed in a previous research [14] to improve the reliability and availability of cloud services as well as the recovery durations in failure circumstances. The authors then developed a framework for distributed applications in a subsequent publication [15] and developed a method for reducing the environment-based service delays for IoT-enabled devices running on the fog-platform utilising an analytical model. In a different paper [16], the authors explored edge-fog computing to address the instability of the MCPS and considering the lengthy lags between their data centres and health devices. By reducing resource consumption costs and ensuring QoS, they also took into account In MCPSs supported by the fog platform, base station association, workload allocation, and virtual machine placement are all possible.

To reduce the travel of data in a fog computing environment, the authors of another paper [17] proposed deploying fog servers. They employed a vector bin packing algorithm as a component of a method to utilise all of the devices' resources. They employed a vector bin packing algorithm as a component of a method to utilise all of the devices' resources. The results showed that their recommended strategy reduced the demand on fog computer resources and cut down on data transportation distance.

In the process of DOG-FPA, difference of Gaussian-based adaptive least mean square algorithm (DoG–ALMS) [24] process is combined with the flower pollination algorithm (FPA) for the optimization and management of the resources from or other resources or clouds. The expected load is distributed to the end-user application via FPA, which also controls the load on every cloud. If a heavy load is predicted, RPM requests the extra resources that are needed. Accordingly, FPA releases some resources if the prediction leads to a load that is more agile. The DoG-ALMS filter's time-series prediction model is used to determine the incoming loads for the upcoming slots. Resources must then be provided for the calculated predicted loads. The variables related to the resources that are now available are then updated for the following time period. This estimation of the variety of jobs and the resources needed for each one aids in triggering the servers fast and with nominal setup time. It provides the persistence characteristics than the conventional FPA.

Hassan et al. (2021) suggested a new service-oriented architecture (SOA) [18] considered as an approach in software design known as, Web Services (WS). It can offer services to other components via a network communication protocol. The management, updating, and to deliver the business processes as (Service Compositions) SCs, which are reconfigured at runtime and contains a collection of WSs that might be called in a particular sequence to handle the clients' requests. The provisions of the Service Level Agreement (SLA) provide that WS to satisfy the needs, selection and composition are important research perspectives expectations. This study offers a useful method that makes use of SMFS. It makes an effort to enhance the WS selection and SC construction, and finally maximise the use of WS resources. The findings indicate that the suggested SMFS approach improves the use of WS resources.

Another research of Hassan et al. (2022) also created the SMO expanded as Spider Monkey Optimization algorithm, and unique model built on the (SMFS [19]) Smart Multistage Forward Search module. Finding the ideal Utilizing the capabilities of cloud computing, a balance between the Virtual Machines (VM) resources, processing power, and service composition abilities was achieved. Among many others, latency is the main issue that cloud computing is now facing.

3.1 Issues Causing High Latency

There are several problems or reasons that make the latency in a cloud environment particularly high. Cloud computing has advanced computer and storage capabilities. The emergence of IoT computing platforms in recent years, however, produced excessive amounts of data was collected and processed using a centralised computing approach with insufficient network capacity, which created a bottleneck because of the volume of data and the challenging latency to the time-critical systems. When different services and applications are sent via the cloud platform for decentralised systems like the Internet of Things, the latency issue only gets worse. To run time-sensitive applications, scale up processing power, and avoid high latency, IoT must utilise cloud resources. The fog platform makes use of a variety of computing tools that are utilised in a variety of areas of daily life, including healthcare facilities, traffic and transportation (roads, trains, shipyards, aviation, etc.), smart homes, as well as the public and private sectors of government and education. Fog component cooperation is crucial for correct and effective functioning [20]. The task time and jitter delays in the cloud, network congestion, and service failures of real-time sensitive apps all impact how quickly a web application loads. This is a hectic operation when it comes to the real-time circumstances. Customers whose enterprises are entirely dependent on network

speed might face catastrophic repercussions as a result of network delays. Users are experiencing unprecedented degrees of dispersed technologies. Users' top priority for smooth operation is a better web experience. Network speed agility is necessary for on-demand cloud services. Everyone's largest problem, latency, must be resolved as quickly as feasible [21]. The time it takes for data to travel from one network-connected device to another, in a broad sense, is the latency of those devices. The time needed for data to travel from the source to the sink through the network link can be viewed broadly as the latency of network-connected devices. These interconnected processing nodes have intrinsic latency, just like in distributed computing platforms, and it differs amongst the three service models for cloud deployment.

3.2 Issues Causing search & service time

In a cloud environment, a number of problems or circumstances take up a lot of search and service time. It is acceptable to create and deploy programmers in web service topologies that make use of the benefits of service-oriented computing. These applications give the creators easy access to a variety of services that offer straightforward and difficult jobs to satisfy the clients' requests, as well as options for reuse. In order to choose the best online services that deliver the required quality of service (QoS) and meet the expectations of the customers, these challenges have been addressed as an important field of research. As a consequence, the workload is divided using the new technique by speeding up the MFS service compositions' response times.

It is a convenient, innovative, practical, and more dependable platform to address the problems with cloud computing. In this method, we emphasize some of the problems that cloud computing end users are frequently dealing with, such as complex latencies and searching time. The Fog-based Spider (FS) in conjunction with the modified Multistage Search (MS) algorithm are provided in our paper work taking into account all of these elements. By dividing them into three channels, the searching time is well used. So, in addition to our prior technique, the Fog-based Spider with Multistage Search (FSMS) strategy is presented in order to shorten the task time of the latency, service, and search processes.

4 PROPOSED MODEL

Using a heuristic technique, the workflow between the various edge nodes in the fog network is improved while delays are reduced (DT) and reaction times (RT) are increased. The basic goal is to find and compute on the closest f-node while minimizing latency between the network's various nodes. The smooth allocation of resources, the availability of services, and the quality of service (QoS) metrics will all improve with a reduction in latency. When it comes to problems with resource optimization in remote computing settings, latency can be a significant impact. Comparing fog computing to cloud computing, the latter offers substantially reduced latency. This technique is taken into the considerations in the proposed module along with the modifications as mentioned below.

Finally, we deliberate the approach of DOG based FPA as a conventional approach. In this process, there is not any providence for latency, any search or service time complexities reduction progress. We implement and incorporate fog based spider using multi-stage search algorithm adapted with the resource utilization for the optimization of web services. Below is a summary of overall process in this chapter.

- This FSMS technique aims to fully utilise mobile-based cloud computing technologies' low-latency and task time usage for resource allocation.
- By utilising Multistage Search (MS) approaches to reduce the number of integrated Web Services, the suggested model improves the web service selection and composition processes.
- It accomplishes this by reducing the response time of the service settings of our prior method and distributing the workload of virtual machines using a load balancer.
- Enhances the scalability and flexibility of service setup and maximises the use of integrated web services' resources in cloud computing technologies based on mobile platforms.
- We do several simulated tests to assess how well our suggested heuristic strategy works. Results of the experiments demonstrate that, when compared to traditional and cutting-edge approaches, our method can perform better in terms of latency control, resource efficiency, and processing time efficiency.

4.1 Fog Spider (FS) Algorithm

The closest nearby possible fog-nodes will be searched via the fog-based spider (FS) algorithm. similar to when a spider traps food in its web inside the perimeter, those nodes near the boundary or close to an edge device can be located very quickly. The edge-request users can able to meet by the accessible nodes. As demonstrated in Figure 1, IoT devices can easily communicate with fog nodes from a variety of local spots to improve searching capabilities and decrease response latency times.

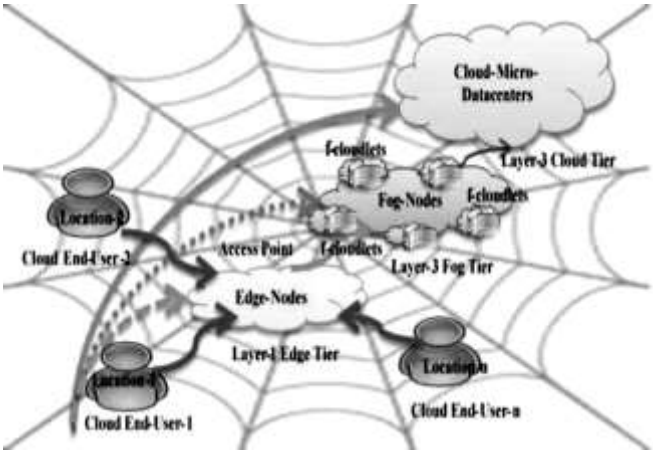

Figure 1 FSA passing through the nearest one

The FS server technique is chosen algorithm is represented as Algorithm 1 in the diagram. To locate the fog server, the shortest paths between each fog device and the service flows are first determined. Dijkstra's method can be used to determine the shortest path between each device and each fog device, according to Lee et al. [21].

Algorithm 1. Fog server algorithm

1. Input:
2. $Resource\_service\_Flows, R = (R_1, R_2, \ldots, R_m)$
3. $Fog\_Devices, d = \{d_1, d_2, \ldots, d_n\}$
4. $minDistance = \infty$

For *each i { 1, ., m }*
    For *each j { 1, ., n }*
        If *(distanceTable(i][j)<minDistance) then*
        minDistance = distanceTable[i][j]
        End If
    End For
assignServiceFlow(Ri, $d_{distance= =minDistance\&\&}$*)*
End For

4.2 Multi-stage search (MS) algorithm

As soon as the service flows are decided, the assignment begins with the service flow that has the fewest fog devices and closest to the multi-stage search process in terms of distance. The final assignment of the fog server deployment location occurs after completion of the subsequent procedures. This final stage can be calculated from MS by calling this function in this FS stage. Thus, both the procedures are combined here, to develop and co-ordinate the FSMS algorithm. By placing the fog server on the fog device nearest to the service flow, the network traffic and service response time can be decreased. In order to distribute the service flows over various fog devices, especially for resource allocations in mobile cloud computing, a fog device with a few assigned service flows is also chosen.

The Dijkstra method is used to calculate the distances between each end device and every fog machine. Additionally, ascending order is used to align the distance tables from the service flows to each fog device. so that the job time complexity is $O(n^2)$. The task time complexity is hence, only marginally impacted by the assignment of the service flows, which just requires a straightforward computation. The FS is heuristically foraging for the optimum local access points with adequate fog serving nodes (Vertices) to improve resource optimization with the shortest possible edge distance between different nodes.

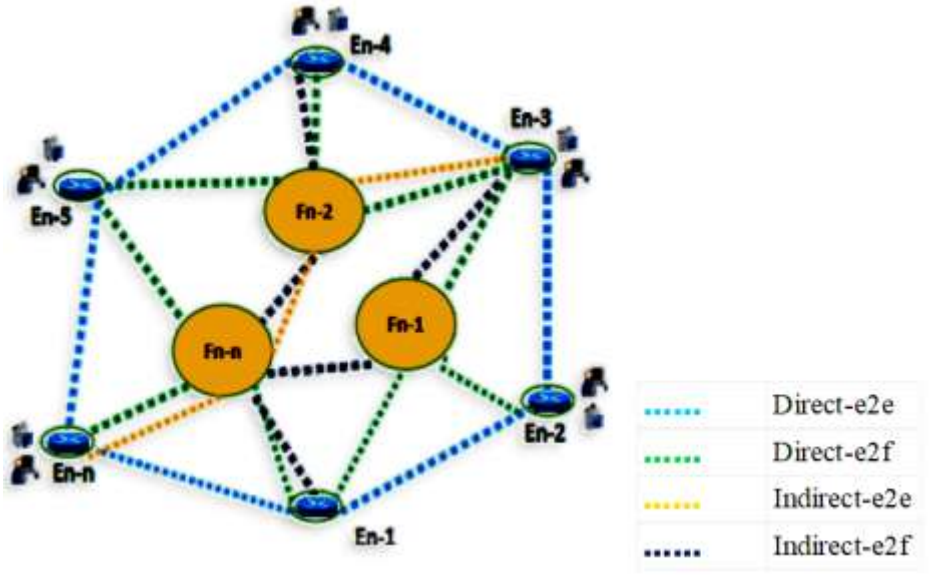

Figure 2 Several edge-nodes proximity connection in Fog-topology

The visiting process between various IoT/edge-nodes and fog-nodes is depicted in Figure 2. As shown in the Algorithm 2 below, one hop is made during the visiting from one node to another. The number of hop counts (nhc) is the separation between the edge device and the fog nodes deployed for each resource. The fewer hops mean lower latency, which makes it easier to find the resources you want. The nodes are geographically scattered throughout the Fog domain, and foglets will fulfill any requests from edge nodes inside the fog layer. The forthcoming algorithm 2, used to determine the minimum distance for this network. It demonstrates that the round-trip time in the fog domain has improved, whereas it is the opposite for cloud users [22]. This provides the better outcomes, such as the minimum angle and minimum distance in the FSMS module for the multi-stage search progress. The distance, d is the minimum distance calculated from algorithm 1.

Algorithm 2. MS algorithm for the distance estimation of hop counts

For *each R ←Assign ServiceFlow*
angle=getAngleBetVectors(R, CurrentResource)
If *(angle < minAngle) then*
minAngle = angle
End ***If***
End For
For *each R AssignServiceFlows*
If *(getAngleBetVectors (R, CurrentResource) == minAngle) then*
else if *(*Mini, < d_{e2e} (n+1)) // *d=minDistance*
nhc=d_{e2e}*(*R_i*),* // direct visit from Edgenode to Edgenode
else if *(*Mini, < d_{e2f} (n+1))
nhc=d_{e2f}*(*R_i*),* // direct visit from Edgenode to Fognode
elseif *(*Mini, > d_{e2e} (n))
nhc=d_{e2e}*(*R_i*+1),* // indirect visit from Edgenode to Fognode
else *(*Mini > d_{e2f} (n))
nhc=d_{e2f}*(*R_i*+1),* // indirect visit from Edgenode to Fognode
End If
End For

The FS reduces the probabilities that data generated by edge devices may cross the border. To cut down service delay time, once the edge-user sends the request, it is routed to the closest accessible fog node rather than the centralized cloud server. The several layers of fog that surround nodes communicate both centrally and via distributed processes. A true fog node controls the other proximity nodes in this central technique, while in the distribution method, all of the fog nodes share control of the nodes and the interaction.

4.3 MS for updating weight modifications

In addition to latency, the length of time the cloud model needs to deliver a service is also taken into consideration. The service composition workflow is organized and managed by the service execution device. Each service composition is made up of a variety of web services, all of which may be changed,

replaced, or controlled while still in use [22] without having any impact on the operational procedures that determine the Quality of Service (QoS) characteristics.

The interconnected web services that make up the service composition determine the QoS. These integrated web services may be impacted by a variety of internal or external factors, including the hosting environment, network, and service upgrade [22, 30], as well as issues with scalability, performance, and the capacity to respond to a large number of synchronous requests while meeting the expected SLA requirements, functionality, and behaviour.

Another problem that might affect the QoS of the service composition is the utilisation of the resources of web services. This concern is motivated by the importance of integrated web services' maximum capacity. A service's maximum capacity is the number of requests it can receive and handle in a single second. The maximum capacity of all linked web services is the same as the highest value web service's maximum capacity.

The MS algorithm used to build the weight calculation is displayed in Algorithm 3. Even though this assessment technique provides the best service to the client, the burden on the web services and service compositions must be taken into account to balance the load on the invoked services during execution. and the highest weight value, ω_i. The number of stages in this multi-search progress can be taken into the consideration of number of hop counts (nhc) from the previous search module.

Algorithm 3. Proposed MS algorithm for the weight calculation

var S; // number of no. of hc in graph G
vari=S-2; // current stage
var E;// all edges connected with j in the upcoming stage(i+1)
***Do** until (no more weight can be constructed)*
***Do** until (i=1)*
nω(i,j)=Min{ω_i(i,E)+ $\omega_i(n+1)$, E},
E in $\omega_i + 1, i = 0, \dots, M-1 \epsilon G$
-i; // prior stage
Loop // stages loop
***Loop // Find next** ω_i*

In this process, the variable *S* denotes the number of stages in FS graph G, which is used to evaluate the best fitness value. It can be considered as multistage search, since it calculates the weight function in all the edges in the next (upcoming) loop, until the prior position is updated. Finally, it is possible to design the service compositions using the best weight value (SC). The Multistage Search (MS) algorithm creates a dependency graph (G) in order to determine the optimal service compositions in the graph. The dependency graph is either modified or regenerated following the upgrading procedure. The revised FS will be searched for new service compositions using the MS algorithm, and the clients will receive the service once more.

4.4 Reduction of task time or service time complexity

One can discover the paths and the potential service compositions in each group using FS dependency graphs and the MS technique. The total number of web services found in the repository is represented by the variable N.

The proposed MFS algorithm is used to paradigm the three groups of service composition. On line Web services connected to each class of clients will be assigned to a global leader, to decrease the task time complexity of Dijkstra's algorithm from $O(N)^2$ to $O(NE_1)^2 + O(NG_2)^2 + O(NP_3)^2$ using the technology of virtualization. The entire number of web services in the excellent class is N_1, the number of web services in the good class with QoS qualities is N_2, and the number of web services with subpar QoS requirements is N_3.

The Virtual Machines (VM) construct a multistage dependency chain for each class group, where the graph nodes correspond to linked FS web services identified in the same class. Based on their action and output, the web services are separated into a number of steps to generate this graph. In order for the service compositions to be saved on the VM and ready for client requests, the MS algorithm is afterwards used to find the optimum routes from the source to the target.

The following equation presents the SMFS' [21] task time complexity:

$$T(n) = (NE_1)^2 + (NG_2)^2 + (NP_3)^2 \quad (1)$$

while Equation (12) shows the proposed MS' model's task time complexity:

$$T_{ji}(n) = (NE_1)^2 + (NG_2)^2 + (NP_3)^2 + (NE_1/ME_1)^2 + (NG_2/MG_2)^2 + (NP_3/MP_3)^2 \quad (2)$$

$$MT_{ji} = min\{C\ T_{ji} \vee T_j \in T, j = 1,2,\dots,n \wedge R_i \in R, i = 1,2,\dots,m\}\ //C{=}n\omega(i,j); \quad (3)$$

4.5 Evaluation of Fitness function

When the construction process of service compositions (SC) is done, FSMS will make the machines with a mathematically uniform distribution (ϵ from a uniform distribution in [0, 1]). The load balancer must be gathered for each and every group j, where MT_{ji}represents the i^{th} machines. Each SMij is initialized as follows:

$$MT_{ji}\ (i,j) = MT_{ji}(minj) + U\ (0,1) + (MT_{ji}(maxj) - MT_{ji}(minj)\ ) \quad (4)$$

It strikes the ideal balance between the processing power, VM resources, and service creation capabilities. Additionally, it increases the effectiveness and efficiency of Web Services' resource use and maximises the resources' capacity for reuse. On the other hand, the extra task time required by the MS approach during execution to look for and build new service compositions in the multi-classes pool is 3(N) 2. By reducing the preceding equation by the number of relevant groups from the web services groups, the next task time equation may be utilised to minimise complexity. The maximum time of completing all the tasks is known to be make span time (MT), which is the time difference between the start time and end time of a sequence of tasks utilizing the resource. The motivation of this combined scheme is to reduce the overall make span. This is applied to allocate the resources to the task based on the QoS constraint with the aim of obtaining reduced time consumption and overall cost in the mobile cloud computing. For every value of calculated MT_{ji}(i,j), we can assess the new fitness function of FSMS using the below equation (5):

$$Final\_Fitness = \begin{cases} \frac{1}{1+MT_{ji}}, & if\left(MT_{ji} \geq 0\right) \\ 1 + \text{abs}\left(MT_{ji}\right), & if\left(MT_{ji} < 0\right) \end{cases} \quad (5)$$

In order to create a service composition, the optimum route to the source must be determined. The initial web service that will be called by the service execution machine will be represented by the path, and so will the last web service in the business process. Additionally, the building procedure involves choosing the web services with the least amount of weight to create the newly optimised weight. Using the FSMS approach, one may do both latency control and service optimization for virtual machines.

5 Result and discussion

MATLAB is used as a simulation platform. The performance of the FSMS in mobile cloud computing is analyzed and the comparisons are made based on the experimental results. Six number of physical or virtual machines with RAM 8 GB and 2TB space for the storage purpose is to be considered. Totally, four numbers of CPUs with a combined 10,000 MIPS capacity shall make up each system. The performance of the suggested algorithm for resource allocation in cloud is employed in these circumstances. The proposed resource allocation method is tested in a simulation environment alongside known resource allocation algorithms including ICSA (Shu et al. 2014) [26], DOG-FPA [24], EMLS (Kessaci et al. 2014) [28], IDEA (Tsai et al. 2013) [27], and DVS (Kołodziej et al. 2014) [29]. The simulative analysis of the performance can be done through the evaluation of parameters such as response time, Resource utilization, and Makespan.

Three performance indicators, such as reaction time, makespan, and resource use, are used for comparisons. The term "makespan" refers to the time required for each resource to execute all of the tasks that have been delegated to them (Kim & Lee 2012 and Zhu et al. 2011) [23 & 25]. The ratio of the physical machine's capacity being employed to carry out incoming tasks to its overall capacity is known as resource utilisation. The average usage ratio is the mean across all resource machines. Reduced reaction time and makespan will result from incoming tasks being completed with the full use of all resources, i.e., complete resource utilisation.

Figure 3 displays the average resource utilisation rate across the range of incoming task counts from 200 to 800. The suggested algorithm FSMS, which was compared to algorithm DOG-FPA, has a higher utilisation ratio (about 80 percent for 800 input tasks). A higher utilisation ratio lowers the high cost and shortens the time tasks must wait in the execution queue.

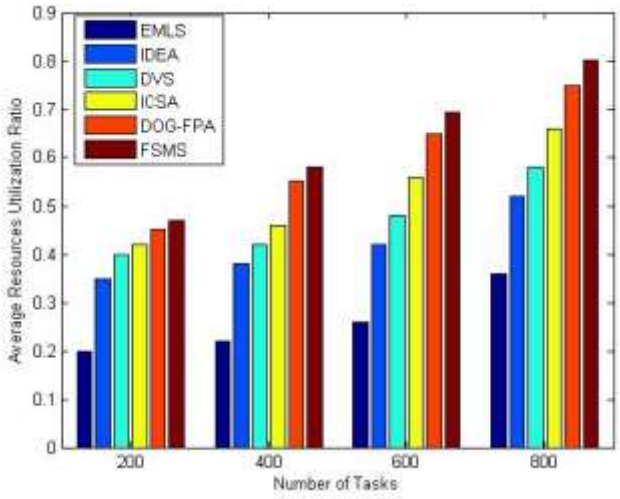

Figure 3. Average resources utilization ratio Vs. Number of tasks

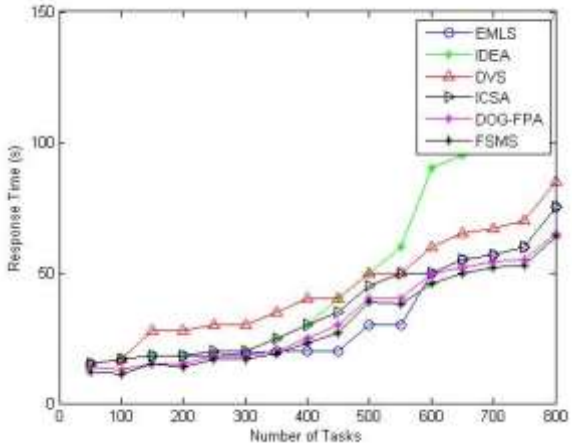

Figure 4. Response time over different tasks

The response time taken of the proposed FSMS algorithm for mobile cloud computing and the formerly implemented DOG-FPA are compared in Figure 4. Response time in DOG-FPA is the interval between the task's proposal of a request and the first response provided by any allocated resource. The more tasks exist, the longer the FSMS takes to respond. EMLS's reaction time at 30 seconds to about 40-second response time is less than the suggested algorithm's, when the number of tasks ranges from 400 to nearly 600. FSMS, however, has an overall response time 65 seconds faster than the other comparing existing algorithms for a greater number of tasks.

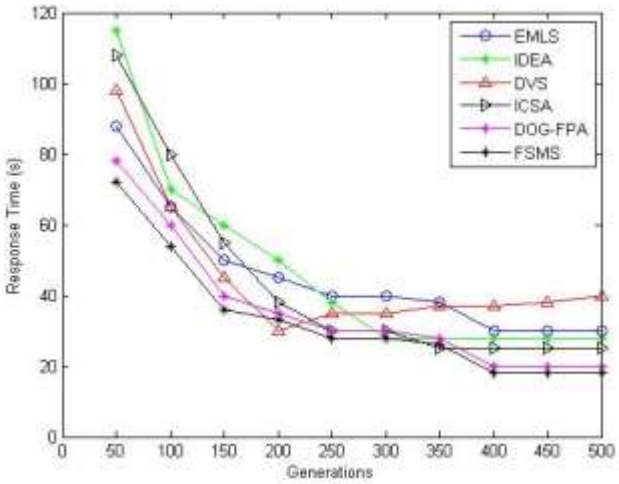

Figure 5. Response time vs. Generations (@ n = 200)

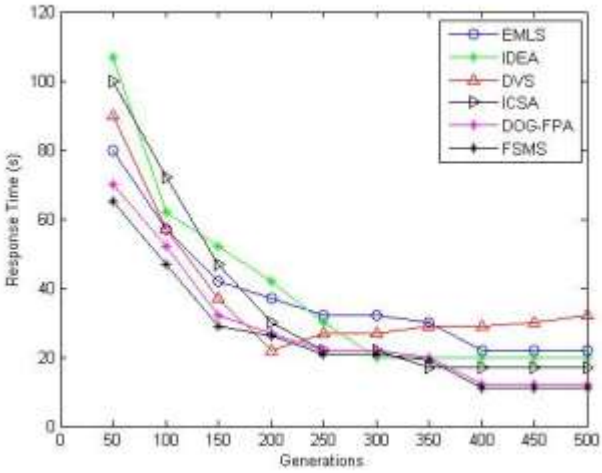

Figure 6. Response time vs. Generations (@ n = 400)

Since resource allocation uses optimization techniques, these algorithms initially produce various combinations of resources and tasks depending on those respective quantities. As a result, given the large number of random possibilities produced, resources would be possibly distributed in the best way. Figures 5, 6, and 7 show the response time graph versus the number of generations for various job n=200, 400, and 500. Changes in response time are examined, according to the number of generations. For n=200, FSMS's response times for various generations are 23s faster than those of the DOG-FPA and other known algorithms. But, DVS and ICSA have a slower reaction time for some responses in some generations.

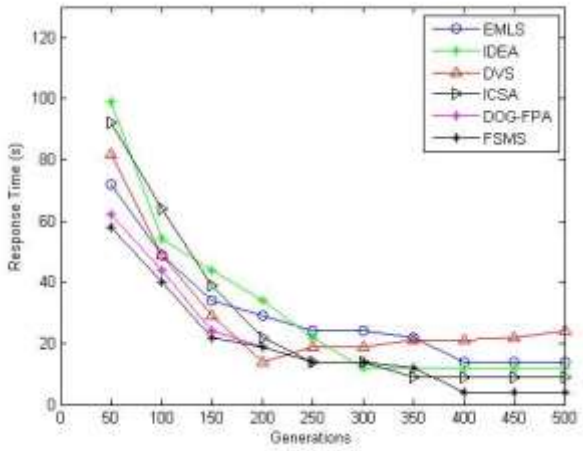

Figure 7 Response time vs. Generations (@ n = 500)

When there are 400 tasks, the proposed module have a reaction time of 19 s, which was even slower than when there were 200 tasks. The suggested algorithm had a reaction time of 15s less when there were 500 tasks. The makespan of the proposed FSMS method and the current DOG FPA are compared in Figure 8. As the number of tasks rises, the value of the makespan increases. The only variation in makespan time between the proposed FSMS, the old DOG-FPA, and ICSA is between 50 and 70 seconds.

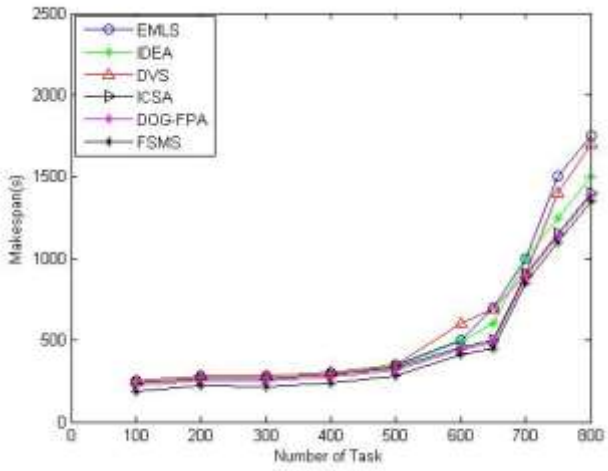

Figure 8. Makespan over different tasks

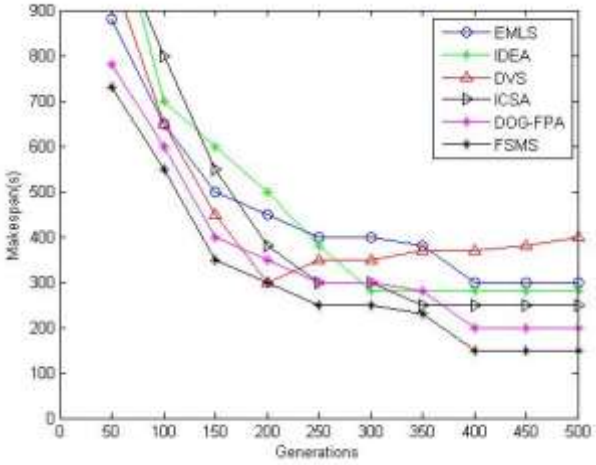

Figure 9. Makespan vs. Generations (n = 200)

The graph showing makespan against the number of generations for different numbers of tasks (n=200, 400, and 500) is presented in Figures 9, 10, and 11. Changes in the make span are examined, according to the number of generations. For n=200, FSMS's makespan is 215 s shorter for various generations than that of existing approaches. But DVS, IDEA, and ICSA have slower reaction times for some responses in some generations.

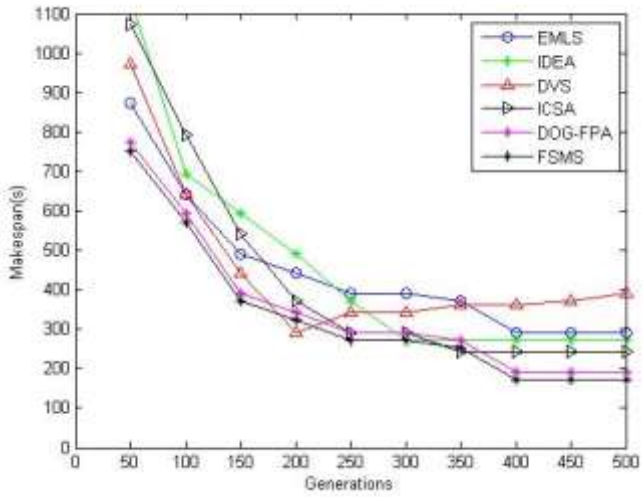

Figure 10. Makespan vs. Generations (n = 400)

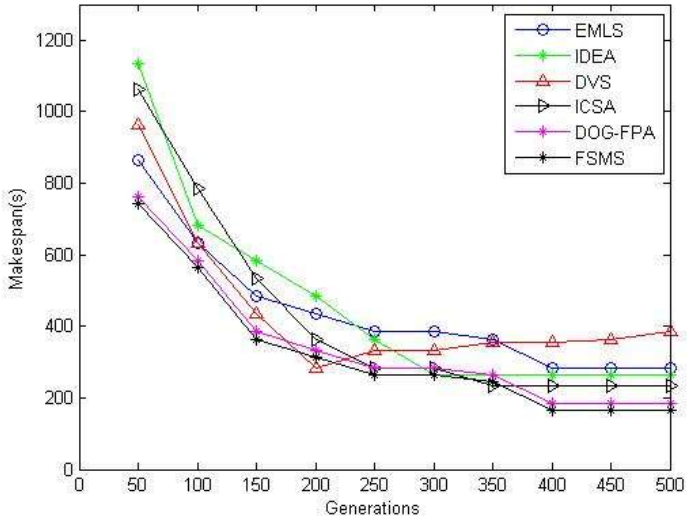

Figure 11. Makespan vs. Generations (n = 500)

The planned FSMS work produced even lower makespan of 200s (same as that of DOG-FPA), when the number of tasks as 400 compared with the incoming tasks as 200. When there are 500 tasks, the suggested FSMS method has a shorter makespan of 190s. By this modified approach, we can achieve both the latency control and service task optimization for virtual machines using the combination of Fog based Spider adapted with the Multi-stage Search management approach.

6 CONCLUSION

In this chapter, we proposed a unique algorithm named FSMS for determining service task time and latency in mobile cloud computing environment, specifically for the resource allocations. Devices IoT for requesting a service or may supply the data needed for dealing out or processing the service. The service flows were delegated to the closest fog device to process the service based on the single service flow. As a result, operating as many services as feasible in a fog computing environment is made possible by the effective utilization of computing resources. Additionally, a faster reaction time may be achieved by cutting down on the distance between each service flow and the fog devices as much as feasible. By cutting down on this distance, the quantity of network traffic can also be reduced. Utilizing a lot of cloud resources and low latency edge resources at the beginning of two phases allowed us to satisfy as many requirements as we could, which served as the cornerstone of our performance. The foundation of our performance was initially taking use of numerous mobile cloud resources and low latency edge resources to meet as many criteria in the first and second stages, respectively. Then, MS tasks that had been transferred from the cloud to the edges were fully used to improve overall performance. Simulative studies are used in conjunction with the experimental testing to demonstrate that our technique outperforms five traditional and modern methods in terms of user satisfaction, processing speed, and resource efficiency. The results are then validated.

In this chapter, we focused on the decisions of both latency control task and service time management in the mobile cloud computing applications. The goal of our future work will be to examine some of the issues such as server maintenance, network failure, buffer size, etc. Additionally, in order to increase

resource economy and processing performance for edge-cloud computing application systems, we will also aim to build an effective caching replacement approach based on anticipating user preferences and user similarity.

DECLARATION OF STATEMENT

The authors declare that they have no conflict of interest.

REFERENCES

1. Mell P and Grance T. The NIST definition of cloud computing, 2011.

2. Ab Rashid D., Ravindran, D. **2019.** Fog Computing: An Efficient Platform for the Cloud-resource Management. *International Journal of Emerging Technologies and Innovative Research*. **6**(2):7-12.

3. Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J. and Jue, J. **2018**. All One Needs to Know about Fog Computing and Related Edge Computing Paradigms: A Complete Survey.1-48

4. Uehara, M. **2017.** Mist Computing: Linking Cloudlet to Fogs. **726**: 201-213.

5. Aazam, M., Huh, E. Fog Computing: The Cloud-IoT/IoE Middleware Paradigm. *IEEE Potentials*. **35**(3):40–44

6. Tang, B., Chen, Z., Hefferman, G., Pei, S., Wei, T., He, H. and Yang, Q. 2017. Incorporating Intelligence in Fog Computing for Big Data Analysis in Smart Cities. IEEE Transactions on Industrial Informatics.13(5): 2140-2150.

7. Jagtap, V. 2014. Survey of Different Swarm Intelligence Algorithms. International Journal of Advance Engineering and Research Development. 1(12): 86-90.

8. Shojaiemehr, B., Rahmani, A. and Qader, N. 2017. Cloud Computing Service Negotiation: A Systematic Review. Computer Standards & Interfaces. 10.1016/j.csi.2017.08.006.

9. Haouari, F., Faraj, R. and M. AlJa'am, M. 2018. Fog Computing Potentials, Applications, and Challenges. IEEE Xplore.399-406.

10. Yousefpour, A., Ishigaki, G. and Jue, J. 2017. Fog Computing: Towards Minimizing Delay in the Internet of Things. 17-24.

11. Abrol, P., Gupta, S. and Kaur, K. **2016**. Analysis of Resource Management and Placement Policies Using a New Nature-inspired Metaheuristic SSCWA avoiding premature convergence in Cloud. *IEEE Xplore*.653-658.

12. Cuong, D., Tran, N., Pham, C., Alam, M., Hyeok S., Jae H. and Choong S. **2015**. A Proximal Algorithm for Joint Resource Allocation and Minimizing Carbon Footprint in Geo-distributed Fog computing. *IEEE Xplore*.324-329.

13. Gorlatova, M., Inaltekin, H. and Chiang, M. **2018**. Characterizing Task Completion Latencies in Fog Computing. *IEEE.*1-13.

14. Kejiang, Ye. & Xu, Cheng-Z. **2018**. Reducing Tail Latency of Interactive Multi-tier Workloads in the Cloud. *IEEE Xplore***.** 162-163.

15. Aazam, M. and Huh, E. **2014**. Fog Computing and Smart Gateway Based Communication for Cloud of Things. 464-470.

16. Hung, P., Aazam, M., Nguyen, T. and Huh, A. **2014**. A solution for optimizing recovery time in cloud computing. *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*. **1**: 1-8.

17. Yousefpour, A., Ishigaki, G., Gour, R.and Jue, J. **2018**. On Reducing IoT Service Delay via Fog Offloading. *IEEE Internet of Things Journal*. **5**(2): 988-1010.

18. Alhadid, I.; Tarawneh, H.; Kaabneh, K.; Masa'Deh, R.; Hamadneh, N.N.; Tahir, M.; Khwaldeh, S. Optimizing Service Composition (SC) Using Smart Multistage Forward Search (SMFS). Intell. Autom. Soft Comput. 2021, 28, 321–336.

19. Tarawneh, H.; Alhadid, I.; Khwaldeh, S.; Afaneh, S. An Intelligent Cloud Service Composition Optimization Using Spider Monkey and Multistage Forward Search Algorithms. Symmetry 2022, 14, 82. https://doi.org/10.3390/sym14010082.

20. Gu, L., Zeng, D., Guo, S. Barnawi, A. and Xiang, Y. **2015**. Cost-Efficient Resource Management in Fog Computing Supported Medical CPS. *IEEE Transactions on Emerging Topics in Computing*. **5**(99): 1-1.

21. Jung-Hoon L. Sang-Hwa C. and Won-Suk K. **2019**.Fog server deployment technique: An approach based on computing resource usage. *International Journal of Distributed Sensor Networks*. **15**(1): 1-19.

22. Karimi, M.; Esfahani, F.S.; Noorafza, N. Improving response time of web service composition based on QoS properties. Indian J. Sci. Technol. 2015, 8, 1–8.

23. Kim, MY & Lee, YH 2012, 'MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server', Computers & Operations Research, vol. 39, no. 11, pp. 2457-2468.

24. Reshmi, R., and D. Shanthi Saravanan. "Load prediction using (DoG–ALMS) for resource allocation based on IFP soft computing approach in cloud computing." Soft Computing 24.20 (2020): 15307-15315.

25. Zhu, R, Qin, Y & Lai, CF 2011, 'Adaptive packet scheduling scheme to support real-time traffic in WLAN mesh networks', KSII Transactions on Internet and Information Systems (TIIS), vol. 5, no. 9, pp. 1492-1512.

26. Shu, W, Wang, W & Wang, Y 2014, 'A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing', EURASIP Journal on Wireless Communications and Networking, vol. 1, no. 1, pp. 64-69.

27. Tsai, JT, Fang, JC & Chou, JH 2013, 'Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm', Computers & Operations Research, vol. 40, no. 12, pp. 3045-3055.

28. Kessaci, Y, Melab, N & Talbi, EG 2014, 'A multi-start local search heuristic for an energy efficient VMs assignment on top of the opennebula cloud manager', Future Generation Computer Systems, vol. 36, pp. 237-256.

29. Kołodziej, J, Khan, SU, Wang, L, Kisiel-Dorohinicki, M, Madani, SA, Niewiadomska-Szynkiewicz, E, Zomaya, AY & Xu, CZ 2014, 'Security, energy, and performance-aware resource allocation mechanisms for computational grids', Future Generation Computer Systems, vol. 31, pp. 77-92.

30. Dongre, Y.; Ingle, R. An Investigation of QoS Criteria for Optimal Services Selection in Composition. In Proceedings of the 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bangalore, India, 5–7 March 2020; pp. 705–710.