SELECTIVE OBJECTIVE OFFLOADING TECHNIQUE IN EDGE COMPUTING USING SOOD ALGORITHM

Pandiyan G

Research Scholar, Department of Data Science and Business System,
SRM Institute of Science and Technology, Kattankulathur, Chennai, India.
pg2874@srmist.edu.in

Sasikala E

Professor, Department of Data Science and Business System,
SRM Institute of Science and Technology, Kattankulathur, Chennai,
India. sasikale@srmist.edu.in

Abstract

Cloud computing serves almost every modern industry by providing infrastructure and computation services. Offloading work to the cloud for computation and storage relieves devices from the need of internal infrastructure. This was later simplified by providing an edge server close to the user. The Edge server provides cloud-like services with low latency and less energy consumption. On the other hand, Offloading task to the edge must be planned in order to optimize server utilization and improve device's Quality of Experience (QoE). The service providers are chosen based on a variety of criteria, and they provide a general optimal server for all tasks. The decision is made based on the server's current state and feedback from the devices. The generic solution reflects the device's service experience and does not provide the best solution for all device criteria. The solution to the stated problem is provided by selective decision-making. The proposed Selective Objective Offloading Decision (SOOD) algorithm adapts a generic algorithm to find the best solution for various devices and server situations. SOOD uses TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) to select servers based on multiple device criteria. SOOD demonstrates that a selective objective-based algorithm outperforms a general objective solution. Prediction of tasks from the device is done by Bi-Directional-Long Short Term Memory (B-LSTM) and the server's level Deep Reinforcement Learning to gather the server information. The simulation results show that proposed model increases the quality of the user's experience individually.

Keywords: Offloading Techniques, Multi-Objective, and Edge Computing.

1. Introduction

Various smart services are continually emerging with the arrival of the 5G network. To finish a plenty of tasks that come from terminal devices, a lot of processing and storage is needed. But the quality of terminal devices limits the number of tasks that can be done. Cloud computing serves the devices by performing both computational and storage processing. Many problems in addressing the processing the tasks from devices to cloud servers are done by mobile edge computing (MEC) [1, 2]. MEC allows terminal devices to not only connect to the network wirelessly but also offload computing duties to mobile edge servers (MES). This MES reduces job completion time, network delay, and energy consumption by reducing the distance between cloud servers and devices. Offloading the task from the device to the appropriate server is the main goal of offloading decision algorithms. Offloading techniques and usage of the resources will have a direct effect on how well the MEC network works.

Most heavy context processing tasks, like AR (Augmented Reality) and VR (Virtual Reality), as well as fast response tasks like self-driving cars and robots, as well as heavy storage analysis tasks like monitoring systems, IoT, and industry 4.0, are sent to edge servers. Offloading decision-making provides a suitable selection process for the task requested from devices. Offloading is determined by a variety of factors, including network path, server load, completion time, priority, and energy of communication, cost, and others. Many algorithms are developed based on the criteria stated above and all provide a common optimal solution for all devices. This is called the Generic offloading technique. Generic techniques analyze a single or combination of some criteria and deliver decision making in the selection of servers. Since the emerging of the criteria is dynamic, it is hard for the Generic technique to work consistently. Considering the edge servers deployed are homogeneous in nature, this will not affect the Generic offloading technique. But in the real world, many heterogeneous servers are available for devices due to the advancing nature of cloud technology.

To address this problem, a selective offloading technique was proposed that considers the heterogeneity of servers available and the different nature of requests from devices. Along with the help of an agent on the cloud side, which aids in decision-making by providing feedback about server history? Frequent request formats are also cached in the SOOD agent to track similar kind of requests and avoid unwanted operations. A multi-objective (MO) problem is formulated for all requests from the device based on their priority and the nature of the task. Generic algorithms like NSGA (Non-dominated Sorting Genetic Algorithm II), AHP (Analytic hierarchy process), TOPSIS, etc. are adopted to solve the MO problem and weighted summation or ranking algorithms are applied to find the right server.

The remainder of the paper is organized as follows: 1) Related works for the proposed model 2) Preliminary problem formulation for selective objective problems 3) SOOD algorithm model and framework proposed 4) Experimental design and comparison of the proposed model to existing algorithms 5) Conclusion and future work.

2. Related Works

The network is made up of edge level, cloud level servers and devices deployed in it. Performance of the entire network is based on fine tuning in all the levels of the network i.e. collaborative decision making on task offloading and processing. First, Prediction is made for the generation of task in device level. Second, to make decision on whether to offload the task to server or process it in the device itself. Last, Selection of server from the available list of service. Recent development in cloud based service provides intense pressure on offloading strategies; predicting and planning scheduling of task and allocation of server incur more stress on devices and to the network.

Adyson m et al [3] adopts genetic algorithm to solve multi objective problem by using random and heuristic initialization. The server placement in edge side and load distribution in server is managed to minimize the response time for deadline task and other operation cost and service availability. Mahbhuba Et al [4] applied Generic algorithm in robotic workflow of industry 4.0 for emergency management service. Cost, Make-span of task and energy are taken as multi-objective problem here and optimized using NSGAII algorithm. NSGAII is modified by making pre-sorted population for initial round and reducing the distance between chromosomes. Laizhong Cui [5] takes energy and latency as multi objective problem and solved it using improved NSGA algorithm. A tradeoff between energy and latency is maintained by providing optimal solution that satisfy all the objectives in the problem.

Gaurav Baranwal [6] implemented modified TOPSIS algorithm for selection of virtual machines from the available fog nodes. Rank reversal in TOPSIS algorithm has been removed to increase the selection process in fog computing. Ranking of the available servers and identifying the application type of the devices are done by monitoring fog node and result will decide the placement of application in upper ranked node. Similar ranking of available nodes is given by Redowan Mahmud [7], instead of TOPSIS a fuzzy logic based ranking is implemented. Fog instances are classified based on their current status and upon the user expectation for their request. User experience is calculated in terms of processing time, service access and resource required, on server side processing speed, proximity or response time and resource availability are considered for rank calculation.

Analytic Hierarchical Process (AHP) is adapted to solve the problem of server selection based on many factors by different researchers. [8] Used two factor for deciding the best server from available servers for particular task. Computation power of the server and the remaining CPU usage are the factor that decides the suitability of the task. Similarly energy of the server, computation time, load of the server also decides the completion of a task. [9] Developed a multi-user communication model for minimizing the time delay and energy consumption of the users. Deep Q Network (DQN) is modelled for this objective by fine tuning the decision-making problem. Reinforcement learning is adopted to adjust the parameter in the decision making through learning from the past offloaded transaction.

The main contribution of the paper is summarized as

1. To find the prioritize factors that are selective for device, which increase the Quality of Experience of the device.
2. Analyze the server capability and status of the server through learning from the previous results.
3. Selective selection of the server based on the ranking of the servers.
4. Updating the experience of the devices after completion of the task to the agent.

Multiple objectives or criteria which decide the offloading decision is adapted in many cases [10]. Many ways to solve this including AHP, GA, PSO, and NSGAII are studied and inference form the result shows that TOPSIS algorithm provides suitable and timely solution for offloading decision making.

3. Task Offloading Problem

In this section the problem in task offloading and its detailed handling in the network is discussed. Task are generated in devices at time interval that are specific to the device nature of operation. Offloading task from the device to edge/device/cloud will be decided according to multiple factors and environment changes. Decision making on considering the different approaches results in achieving the efficient modelling of the network. Task generation in devices will be considered as important factor for offloading decision.

Let's consider the important factors that affect the overall working of the offloading model. Prediction of task that will be arise in future is one of the key factors while formulating offloading technique. Service provider prepares the resources in advance of incoming task to reduce the time to complete the task. Offloading decision based on server status of all providers and upcoming status of the server prediction will also increase the result of offloading problem. Continuous monitoring of the server status and updating it in offloading agent will increase the accuracy in offloading. Selective decision for different kinds of request will eliminate the error in offloading. Considering this factors and modelling the network operation provide overall solution for all problems in cloud edge computing.

3.1. Prediction of task using B-LSTM

The selective objective of the device will decide the quality of the experience of the device, which can be predicted for the device so that offloading decision can be made in advance of the request from device. B-LSTM is implemented for predicting the upcoming request type from the device based on their request history [11]. Device request are given as input for SOOD algorithm for deciding the ranking calculation.$C = \{C_1, C_2, C_3, \dots, C_n]$ Are the criteria for the device and based on that ranking is calculated in SOOD algorithm. Let $C'$is the predicted criteria of the device requests on next turn and C is the actual criteria of the request made at time t. If, $C' = C$ then the offloading decision already made on agent is applied which will increase the overall response time of the request. If, $C' \neq C$ then C is taken as the input for the decision making and offloading decision is given based on the ranking of the algorithm.

B-LSTM takes two hidden layers for increasing the accuracy by considering both the past and future context of the prediction data. Forward hidden layer H_t^f and backward hidden layers H_t^b takes the input in ascending and descending order respectively and the output of this layer is combined and forwarded to next layer.

Forward layer is represented as

$$H_t^f = \tanh\left(W_{xh}^f x_t + W_{hh}^f H_{t-1}^f + b_h^f\right) \tag{1}$$

Where H_{t-1}^f is the forward hidden state input, W is a weight matrix (W_{xh}^f is a weight connecting input (x) to hidden layer (H)) in forward direction, b_h^f is a forward bias vector, and tanh is an activation function of the hidden layer.

Backward layer is represented as

$$H_t^b = \tanh\left(W_{xh}^b x_t + W_{hh}^b H_{t+1}^b + b_h^b\right) \tag{2}$$

Where H_{t+1}^b is the backward hidden state input, W is a weight matrix (W_{xh}^b is a weight connecting input (x) to hidden layer (H)) in backward direction, b_h^b is a backward bias vector.

Output is calculated as

$$y_t = W_{hy}^f H_t^f + W_{hy}^b H_t^b + b_y 2 \tag{3}$$

Where $W_{hy}^f$the output of forward is weighted matrix and W_{hy}^b is output of backward weighted matrix. Learning rate is set as 0.001 as at 0.0001 the convergence rate is slower and longer time is taken for obtaining optimal, at 0.1 the fluctuation of loss is high which fails to converge. ReLU will be used as an activation function between the input layer and the hidden Layer. ReLU is less computationally intensive than sigmoid or *tanh*, and it will speed up the training for initial data.

3.2. Server Analysis using Deep Reinforcement Learning

Edge servers are of heterogeneous in nature and they provide service with different levels. Not all the servers be able to provide similar solution for the needs of device, that's why offloading decision are made to find the appropriate one. The state of the servers are monitored by the cloud agent by continuous analysis of its performance on every response of the request by devices. Deep reinforcement learning learns the state of the servers including load prediction, latency, type of service, cost, accuracy, waiting time, success rate, Etc. This states $E = \{E_1, E_2, E_3, \ldots E_n\}$ gives the value for rank calculation in SOOD algorithm. The cache in cloud agent contains this state and they are forwarded to edge agent and devices. Based on the values future decision making on offloading for the devices are made.

The DRL contains agents, environment, state, action and reward function. Constant learning of the environment and decision made as per policy is done by agent. The state, $s \in S$ at every time t is monitored by agent and the action taken. The policy which determines the agent for decision-making is given as $U: S \rightarrow a.$ the reward given after the action made is given as $r_t = R(s, a)$ and the state after action is given as $S_t \rightarrow S_t + 1$. Discount factor is sum of rewards and previous state achieved by agent. Which is expressed using the Bellman equation. It is expressed as

$$\mu^*(s) = \arg\max_a Q^*(s, a) \tag{4}$$

Which is expressed using the Bellman equation. It is expressed as

$$Q^*(s, a) = E\,[R(s, a) + \gamma \max_{a'} Q^*(s', a') \tag{5}$$

DRL is trained with the value of decision from the TOPSIS agent. The reward and penalty is calculated for the selection of the server. This will be reflected in the values of the server in the cache.

3.3. Generic Offloading problem

The offloading decision is calculated based on the feedback from the system about the state of the servers and the request from the device. Decision-making using the server state and general objectives of the devices is done in previous works [12] [13]. Metrics like Accuracy, Access rate, Drop rate, Energy, Etc. are taken while formulating the ranking process in existing works. This will provide solution for all requests similarly irrespective of individual priorities. Ranking algorithm like TOPSIS, AHP, etc. uses these metrics and weights based on predefined values of the criteria. General solution increases the performance of the entire network while individual clients suffer from experience on the service. Weight of the criteria of the ranking is based on fixed values or variable values for all request at a time W = a, b . . . x.

Here values are set based on the overall requirement of the network. One model is set as a>b>...>x. as has high priority then b and so on. Another type is a=α1, b=α2. . . etc. here a, b. . . x has given a different values based on their importance in ranking. Either way has a drawback; that is, it is fixed for all requests from the devices in the network, which are set to increase or decrease an individual or set of network performance metrics or device experience metric.

4. Proposed Model

4.1. System Architecture

The system architecture of the proposed model is given in Fig. 1. The model consists of mainly two objectives. 1. Analysis of properties of the edge server and the device by DRL in cloud agent. 2. Selective objective based offloading decision for the devices request using SOOD algorithm. Devices request for service is of different type for each devices. Quality of Experience of the device has direct impact on request priorities. A device request with priority for low delay and storage cost needs server with this attributes to process its request. This is selective objective for that device and offloading decision has to be made accordingly.

4.2. Selective objective offloading

The problem is to select the optimal server upon many available servers for particular priority of the requests of the device. Edge server is considered here as alternative for multi-criteria decision making and ranking. {E_1, E_2 . . . En} are set of available servers for the device and {C_1, C_2 . . . Cn} are the criteria of selection, here it is priority of the devices for offloading. The offloading decision for mobile devices is considered as a Multiple Objective problem, selecting appropriate server is based on the factors that devices considered as typical one for their QOE. Different solutions have been provided for different users to increase the chance of successful servicing and also to predict future request.

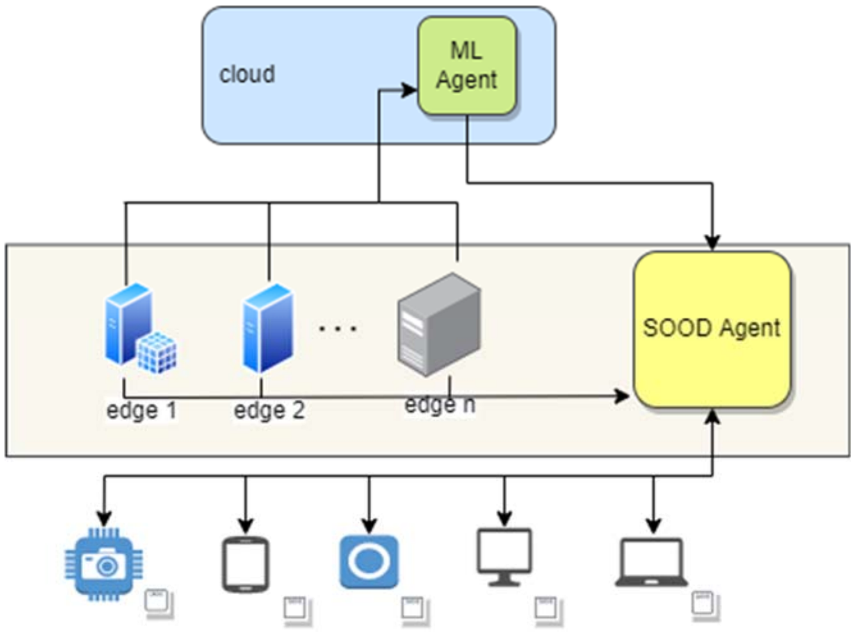

Fig.1. Architecture of Proposed Model.

4.3. SOOD Algorithm

SOOD Algorithm

```
Input: request $r_i$ from devices
        Cache from agent $C_a$
        Cache from devices $D_a$
        Feedback from cloud agent $CO_a$
        Server List $SL = \{E_1, E_2, .... E_n\}$
Result: offloading decision
1.  $i = 0$ ;
2.  While $i \neq 0$ do :
3.      Get factor and priority of $r_i$ as $FAC(r_i)$
4.      if $FAC(r_i) \in D_a$:
5.          Select the $E$ from the server list $SL$
6.      else:
7.          if $FAC(r_i) \in C_a$:
8.              Select $E$ from $SL$
9.          else:
10.         Run TOPSIS Ranking
11.         Select $E$ from $SL$
12.         Save $FAC(r_i), S$ in $D_a, C_a$
13. Forward feedback($r_i, E$) to $CO_a$
14. $i \rightarrow i + 1$
```

Agent in edge level, performs the selection of server for the device based on their predicted priorities of task. This selection process is a chain of decision on every level, first the device is checked with its cache D_a for server selection based on most recent suggestion. t ′ is the time limit which determines the confirmation of previous suggestion, as the status of the server changes frequently only continuous task request are taken for decision D_a. If the selection is made on Device level, then direct offloading of the task to the server is made. If t ′ is timed out then it sends to the edge level for offloading decision. C_a is the cache that is available in the agent in edge level, this is where Modified TOPSIS algorithm runs for finding offloading decision. Similar to D_a, a time restriction is made for utilizing the cache in edge agent, if it timed out then fresh decision is made for given attributes from the device. Fig.2. explains about the operation of Offloading Agent. Cache from the cloud CO_a is updated with the status of the servers along with the trust value. Updating of the status to the agent in edge is done on timely interval and on demand request made. D_a is updated from both cloud agent and its own feedback and statics which is used to decide the server selection in TOPSIS algorithm. Server selection is made and the performance is updated to the CO_a and in D_a.

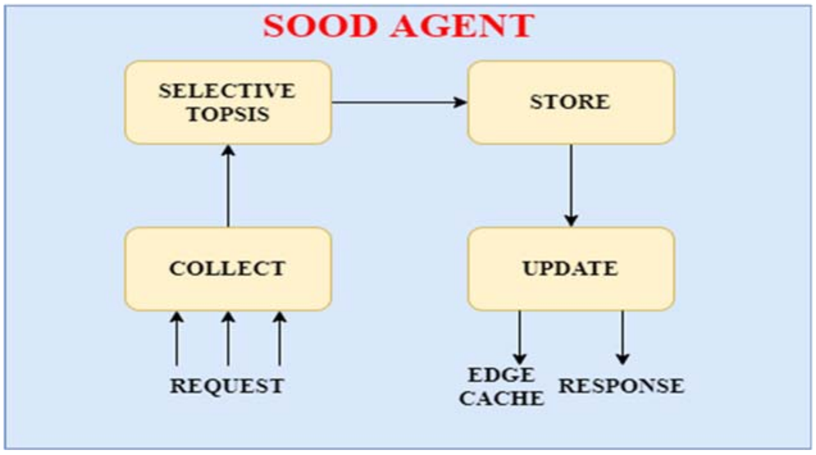

Fig.2. Operation of the SOOD Agent.

4.4. Modified TOPSIS

The TOPSIS method works on determining a preference index in order to rank alternatives by computing the shortest Euclidean distances to both positive and negative ideal solutions at the same time. It is based on the idea that the chosen alternative should be the closest to the positive ideal solution and the furthest away from the negative ideal solution.

The TOPSIS process is carried out as follows:

Step 1: Identify the priorities of the devices and then construct a m x n matrix for that particular request. Edge devices are listed as $E = E_1, E_2, \ldots E_n$. The matrix consists of values of the edge servers E received from the cloud agent at time t of the request made. Let $P = P_1, P_2, \ldots P_n$ are the priority of the request made by device. Then E X P matrix is constructed with p values of each server. E is the alternatives and P is the criteria of the multi criteria problem.

$$E\,X\,P = \begin{pmatrix} EP_{11} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & EP_{nn} \end{pmatrix} \tag{6}$$

SERVER	Factor 1	Factor 2	.	.	Factor n
E1	X_{11}	X_{12}	.	.	X_{1n}
E2	X_{21}	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
En	X_{n1}	.	.	.	X_{nn}

Step 2: The normalized matrix is constructed by using the formula

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \tag{7}$$

Beneficial values and non-beneficial values are given for ranking process, beneficial values indicates higher the value greater is the selection, whereas non-beneficial value have lower the value better the selection.

Normalization of the attributes gives the values for calculating the weightage of the server which in turn is used for ranking of the server.

Step 3: Weighted matrix is constructed from the weight of the priority and normalized value of the priority. $W_{ij} = w_p \times r_{ij}$. w_p Is the weight of the priorities given by the requesting device. Here weight is set based on priority i.e, higher the priority, higher the weight similarly, lower the priority, lower the weight. Consider the priority P of device as {p_1, p_2, p_3, p_4} then weight is set as {4, 3, 2, 1} as p_1 have high priority its value has been set as higher while considering the ranking process, and p_1 is set to 1 as it has low priority.

Step 4: Determining the ideal solution matrix of positive and negative ideal solution by using this formula:

$$A^+ = \left\{\left(max\mathrm{v_{ij}} \mid \mathrm{j} \in \mathrm{J}\right), \left(\underset{\mathrm{ij}}{min} \mid \mathrm{j} \in \mathrm{J}'\right), \mathrm{i} = 1,2,3,\dots,\mathrm{m}\right. \\ = \{\mathrm{V}_1^+, \mathrm{V}_2^+, \mathrm{V}_3^+, \dots, \mathrm{V}_n^+\} \tag{8}$$

$$A^- = \{(min\mathrm{v_{ij}} \mid \mathrm{j} \in \mathrm{J}), (max\mathrm{v_{ij}} \mid \mathrm{j} \in \mathrm{J}'), \mathrm{i} = 1,2,3,\dots,\mathrm{m}\} \\ = \{\mathrm{V}_1^-, \mathrm{V}_2^-, \mathrm{V}_3^-, \dots, \mathrm{V}_n^-\} \tag{9}$$

Ideal positive solution is the best value i.e. maximum value for beneficial criteria and minimum value for non-beneficial criteria. Ideal worst solution is worst value i.e. Minimum value for beneficial criteria and maximum value for non-beneficial criteria. Ideal solution is calculated based on the impact of the criteria in ranking process. Edge selection process includes many criteria like processing time, delay, cost, waiting time etc., from the request of the device beneficial and non-beneficial criteria is identified. Processing time criteria is considered as non-beneficial, server with low value has high chance to be selected, in contrast CPU frequency is beneficial criteria, and server with high value has high chance to be selected. So processing time, cost, delay etc. are non-beneficial whereas CPU speed, memory, available resource, type of service are beneficial service. Ideal positive solution is calculated as max for beneficial and min for non-beneficial.

Step 5: Calculating separation - Next step is to calculate the distance between ideal best / worst solution for the servers. Sum of square of the difference between ideal best/worst value and server attribute is calculated and square root of the value gives S^+/S^- value. This ideal distance is used to calculate the score for ranking process.
a. S_i^+ is an alternative distance from the positive ideal solution is defined as:

$$s_i^+ = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_i^+\right)^2} \tag{10}$$

Where i = 1, 2, 3... m.

b. S_i^- is an alternative distance from the negative ideal solution is defined as:

$$s_i^- = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_i^-\right)^2} \tag{11}$$

Step 6: Scoring calculation is done by finding from the ratio of S^+ or S^- and sum of S^+ and S^-. This score decides the ranking process in final step. Scoring value decides the ranking process of the server. It consider all the attributes of the server and comparison of the values among the server to find the best one among them.

$$S = \frac{S^+}{S^+ + S^-} \tag{12}$$

Step 7: Ranking the edge server depends on the attribute, criteria and its weightage. Weightage is given by the device and its request. Ranking is based on the score which selects the servers for offloading processes process.

5. Result Analysis

The environment setup is done by using Edgecloudsim package in eclipse 2021. Edgecloudsim is a simulation setup done in java in eclipse environment that supports edge cloud network. The network is set with one cloud layer where the cloud agent runs and edge layer is deployed with 8 edge servers and device layer with multiple devices. The configuration of the servers and devices are shown in Table I. The tasks are generated at random Poisson distribution from devices range from 200 to 1000 devices. Each device is categorized in 4 general manner as Infotainment, Heavy computation, Augmented Reality, Health app. Tasks are generated in distribution according to their type and at random manner. Health app consider the priority as storage, storage cost, del while heavy computation gives more priority to CPU usage, delay, no of CPU, resource cost. Each devices is given priority in their type in distribution.

Let T1 is the new task requested from the md1 with fac () = {f1, f2, f3} which is predicted by the B-LSTM in edge agent. The goal of the proposed model is to allocate service provider for the task request which satisfy its priorities.

Factors	Metrics
No of core	16-20
Mips	60000 - 100000
Storage	10tb -20tb
vms	8-14
Os	linux
Architecture	X86
CostperBw	0.1-1.0
CostperSec	30-50
CostperMem	0.05-2.0
CostperStorage	0.1-2.0

Table 1. Configuration of Server

Selection of server with general factors won't satisfy all request types, in order to satisfy individual request, Algorithm finds appropriate server based on ranking process. Let consider the scenarios where different request type from the devices and selection of server based on their constraints is done.

5.1. Scenario

5.1.1. Scenario 1:

The request from the device is monitored by the agent and priorities are set by the devices for future service with the edge computing. One scenario is, if the priority was predicted as Type of service, low Cost, low delay, as it process some infotainment application. This are the preferences of the request and the quality of the experience is high when it satisfy this criteria. So weightage of the criteria is set based on that as delay=4, CPU =3, cost =2, and service type =1. The offloading decision on selection of the server is done by applying proposed TOPSIS algorithm and based on the ranking selection of the server for service. Table II shows the ranking for Scenario 1.

Criteria → high type of service→ cost → delay low → failure rate.

Table 2. Ranking for Scenario1.

Edge server	Type of service	Cost	Delay	Failure rate	Rating
E1	1	2	.24	.56	3
E2	1	3	.45	.12	4
E3	0	4	.31	.45	5
E4	0	1	1.2	.24	2
E5	1	3	2.5	.34	1

5.1.2. Scenario 2:

Another scenario is application which are concerned about Accuracy, Trust, and Type of service and Delay. Application like scientific or research oriented concerns about high accuracy and service oriented server for its request. In scenario like this selection of server will take in to account about their priority and appropriate server has to be allocated for them. Table 3. Shows the server selection for this scenario with same edge servers. Unlike previous scenario where E5 server provides better solution, here E2 server is selected for the service for this request criteria.

Criteria →accuracy →trust → type of service →delay.

Edge server	**Accuracy**	**Trust**	**Type of service**	**Delay**	**Rating**
E1	.95	2	1	.34	4
E2	.97	3	1	.56	1
E3	.89	4	0	.12	3
E4	.94	1	0	.45	2
E5	.96	3	1	.23	5

Table 3. Ranking for Scenario2.

5.2. Result Comparison

5.2.1. QOE of Device

Selection of servers is finalized by ranking of available servers. The service quality is increased by selecting the appropriate server. Ranking is proposed model is noticed with high accuracy than other model. Fig 3. Shows the comparison of accuracy of the selection process. Quality of Experience is calculated by comparing expectation with actual result. Fac() is given by devices at the time of request, once they are offloaded with this factors as deciding criteria for selection. The ratio of received criteria with expected factors gives the quality of experience for the device. Proposed algorithm achieves 98.4% on user satisfaction compared to 96.61%, 94.2% and 92.34% by other models.

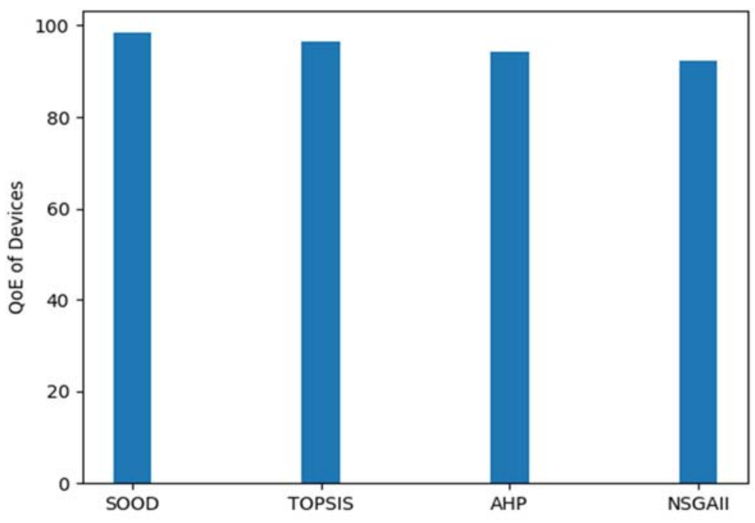

Fig.2. QoE of Devices for different offloading algorithm.

5.2.2. Offloading Decision Time

Each request have their own priorities and weights and on this basis the server is selected. Proposed model takes that into consideration and with the help of the agents it decides the server for the request from device. SOOD algorithm reduces the take taken for the decision making by utilizing the cache that is available in device and in agent. The decision is made fast by using the cache which will have previous request-service feedback. So extra time is taken only if new type of request is arrived. fig4. Shows the comparison on time taken by the algorithms for offloading decision making.

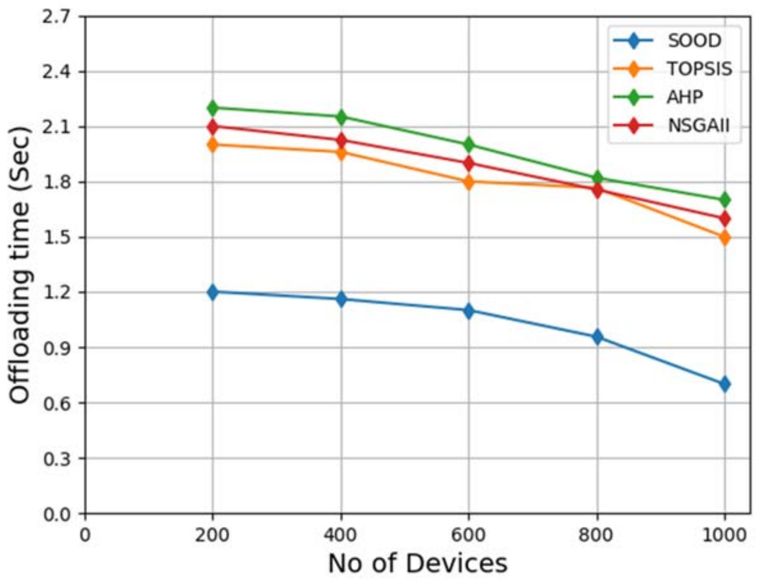

Fig.3. Time taken for offloading decision for number of devices.

5.2.3. Failure Rate

Failure of the service happens when the server fails due to overload or when time exceeds the threshold value. Also selection of server leads to failure of service, as selecting incompetent server always end in service failure. Proposed model not only selects based on the present state of the server but also it considers the feedback of the server on previous service. The failure rate of the service is reduced a lot which is achieved by feedback from the agent. Learning algorithms provides the accurate information about the states of the servers in past and in present. Fig 5. Shows the failure rate comparison of proposed model with existing models. Proposed model SOOD algorithm reduces the failure rate notably.

5.2.4. Ranking Accuracy

Ranking accuracy of a multi objective problem depends upon how extent they satisfy the constraints of the input. Selection process involves selecting the server which almost satisfy every need of the request. A balance between the attributes of the request has to be maintained so that it will cover every attribute and also top among the available outputs. Proposed model of ranking considers all the attributes and their priority based weight and finally picks the best one among them. Fig 6. Shows the comparison of ranking accuracy among models.

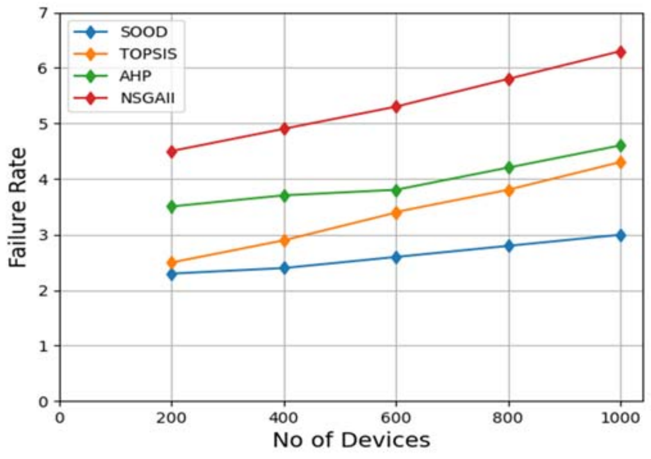

Fig. 4. Failure rate comparison of different models.

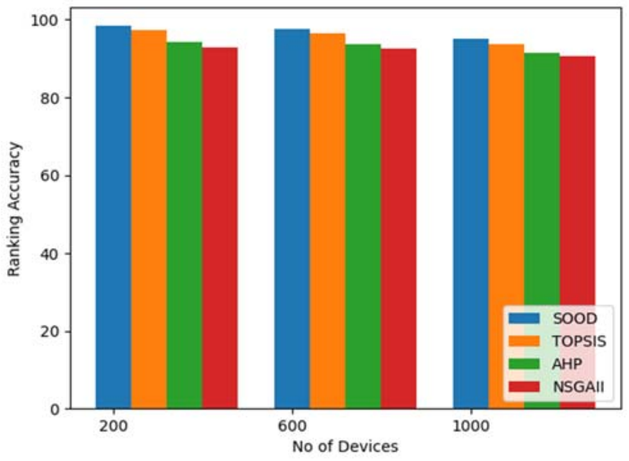

Fig. 5. Accuracy on ranking the server.

6. Conclusions

Allocation of the task to server will satisfy the requirements of the requesting device based on the learning model. This process involves the feedback from the cloud agent and from the device's task prediction. Selective objective based offloading decision making increases the QoE of the devices. Prediction of upcoming request and its criteria are done by B-LSTM which reduces the offloading time. Proposed model incorporates feedback and multi objective problem to identify appropriate service provider. In future work, dynamic server availability is taken in consideration for modelling the offloading problem.

Acknowledgements

The authors would like to thank everyone, just everyone!

References

[1] Kumaran, K. & Sasikala, E, (2021) "Learning based Latency Minimization Techniques in Mobile Edge Computing (MEC) systems: A Comprehensive Survey", *2021 International Conference on System, Computation, Automation and Networking (ICSCAN)*, pp. 1-6.

[2] Saranya, G. & Sasikala, E, (2021) "Offloading Methodologies for Energy consumption in Mobile Edge Computing", *2021 2nd International Conference on Smart Electronics And Communication (ICOSEC)*, pp. 832-838.

[3] Maia, A., Ghamri-Doudane, Y., Vieira, D. & Franklin de Castro, M. (2021) "An improved multi-objective genetic algorithm with heuristic initialization for service placement and load distribution in edge computing". *Computer Networks*. 194 pp. 108146.

[4] Afrin, M., Jin, J., Rahman, A., Tian, Y. & Kulkarni, A. (2019), "Multi-objective resource allocation for Edge Cloud based robotic workflow in smart factory". *Future Generation Computer Systems*. 97 pp. 119-130.

[5] Cui, L., Xu, C., Yang, S., Huang, J., Li, J., Wang, X., Ming, Z. & Lu, N. (2019). "Joint Optimization of Energy Consumption and Latency in Mobile Edge Computing for Internet of Things". *IEEE Internet of Things Journal*. 6, pp. 4791-4803.

[6] Baranwal, G., Yadav, R. & Vidyarthi, D. (2020) "QoE Aware IoT Application Placement in Fog Computing Using Modified-TOPSIS". *Mobile Networks and Applications*. 25, pp. 1816-1832.

[7] edowan Mahmud, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2019. “Quality of Experience (QoE)-aware placement of applications in Fog computing environments”. *J. Parallel Distrib. Comput*. 132, pp. 190–203.
[8] M. Yang, H. Ma, S. Wei, Y. Zeng, Y. Chen and Y. Hu,(2020) "A Multi-Objective Task Scheduling Method for Fog Computing in Cyber-Physical-Social Services," in *IEEE Access*, vol. 8, pp. 65085-65095.
[9] M. Zhang and L. Liu, (2015) "Evolutionary Algorithm with AHP Decision-Making Method for Cloud Workflow Service Composition," *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 339-346.
[10] Xu, X., Li, Y., Huang, T., Xue, Y., Peng, K., Qi, L. & Dou, W. (2019) “An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks”. *Journal of Network and Computer Applications*.
[11] Yu, S., Wang, X. & Langar, R. (2017) “Computation offloading for mobile edge computing: A deep learning approach”. *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. pp. 1-6.
[12] Huang, L., Bi, S. & Zhang, Y. (2020) “Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks”. *IEEE Transactions on Mobile Computing*. 19, 2581-2593.
[13] Chen, X., Zhang, H., Wu, C., Mao, S., Ji, Y. & Bennis, M. (2018) “Performance Optimization in Mobile-Edge Computing via Deep Reinforcement Learning”. *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. pp. 1-6.
[14] Wang, J., Liu, K., Ni, M. & Pan, J. “Learning Based Mobility Management under Uncertainties for Mobile Edge Computing”. *2018 IEEE Global Communications Conference (GLOBECOM)*. pp. 1-6.

Authors

Pandiyan G Received his B.E in Computer Science Engineering in the year 2009 and M.Tech in Computer and Communication from Anna University in the year 2012. He is pursuing Ph.D. degree from SRMIST. He have more than 7 years’ experience in teaching and 2 year in industry. His research interest includes Cloud computing, wireless sensor network, Machine Learning.

Sasikala E Received her B.Tech degree in Information Technology in the year 2006 and M.E degree in Computer Science Engineering in the year 2010 and Ph.D. degree from Anna University in the year 2017. She is working as Professor in the department of Data Science and Business Systems at SRMIST. She has authored or co-authored more than 20 reputed journal and conference papers. She has acted as a principle investigator in In DRDO project. Her research interest includes wireless sensor network, security, intelligent techniques.