Journal of Electrical Engineering & Technology (2023) 18:3897–3909
https://doi.org/10.1007/s42835-023-01474-3

ORIGINAL ARTICLE

Dynamic Data Replication and Scheduling Using Fuzzy-CSO Algorithm for IoT-Clouds

M. Saranya[1] · **R. Ramesh**[1]

Received: 9 November 2022 / Revised: 15 February 2023 / Accepted: 9 March 2023 / Published online: 28 March 2023

Abstract
Data replication and task scheduling are two strategies to enhance the performance of data-intensive applications. One of the main issues in the Internet of Things (IoT)-Cloud scenario is uploading data from the sensor gateways and replicating it across multiple cloud data centres (DCs) for high availability. To avoid such problems, there is a need to adaptively determine the number of replicas and their optimum locations. Although data replication ensures availability and reliability, keeping many copies of each data will increase storage space use. To overcome this problem, a minimal number of replicas need to be maintained for these files. Most of the existing works consider the system as non-faulty, but in real-time, various faults may occur at every data centre (DC). Hence, the main objectives of this research work are to adaptively determine the number of replicas and their optimum locations, as well as to design a fault-tolerant scheduling algorithm for IoT-based Cloud. This paper deals with the design of dynamic data replication and scheduling framework using the Hybrid Fuzzy-CSO algorithm for the IoT-Cloud. It uses the Cat Swarm Optimization (CSO) algorithm to find the optimal locations for replications. The fitness function is derived from the distance between the main DC and the other DCs. A Fuzzy logic decision model was designed to determine the optimal number of replicas. During task scheduling, data replication was performed in the selected replication points and scheduled accordingly. The experimental results have indicated that the proposed Fuzzy-CSO framework attains minimum data transfer time, minimum response delay, and higher bandwidth utilization than the existing algorithms.

Keywords FLD · IoT · CSO · Fuzzy · Replication

Abbreviations

IoT	Internet of Things
GW	Gateway
DC	Data center
CSO	Cat swarm optimization
FLD	Fuzzy logic decision
CC	Cloud computing
DR	Data replication
QoS	Quality of service
PSO	Particle swarm optimization
FOP	Fault occurrence probability
TEC	Total energy cost
TSC	Total storage capacity
ABO	Artificial butterfly optimization
VM	Virtual machine

✉ M. Saranya
msaranyaphd04@gmail.com

R. Ramesh
rramesh@annauniv.edu

[1] Department of Electrical and Electronics Engineering, College of Engineering, Anna University, Chennai 600025, India

1 Introduction

IoT occurrence is making a domain of billions of associated devices producing massive quantities of data stuff. IoT, which comprises devices like sensors, actuators, and GWs, is capable of a variety of novel applications by permitting several devices for association and communication through the Internet deprived of human interference. The IoTs have provided certain extended uses and chances that comprise smart grids for augmenting the dependability and efficacy of power supplies [1]. Since IoT devices are not fortified with widespread storage, their data needs to be stowed outside them. Given that transmitting all of this data to one location would make data analysis difficult, dispersed data storage over numerous geographically dispersed clouds was chosen. Cloud computing (CC) and Big data play important roles in data storage and the scope of the study. CC has made an additional novel method of executing and deploying the applications identified on the Internet [2].

In IoT-cloud infrastructure, data eruption is predictable to ensure the capability of cloud DCs. Besides, the transmission of the entire data to an integrated place would make condensing data examination problematic. Hence, disseminated data storage is done amongst numerous geographically disseminated mini-DCs [3]. One of the principal investigation issues in IoT-Cloud is uploading data from the GWs and replicating such data in manifold DCs for providing high accessibility. For the determination of high accessibility, replication of this data requires proficient application [4]. Replication involves the procedure of offering diverse models of similar ability at diverse nodes. The data replication scheme prudently permits access to data and provides earlier access to the records needed by cloud jobs. It also ends in condensed bandwidth depletion [5].

An effective task scheduling method is needed for boosting the execution time. Current task scheduling algorithms are primarily concerned with task-resource specifications, Memory access resources, runtime, and operating costs. Data replication (DR) and scheduling are two production methods that can augment data-based applications. In one way, programming eliminates replication of the data records essential for them with remote access to the records. Tenuous retrieval of these data records needs more time than retrieving directly, hence doing programming devoid of replication involves overhead of data access time. Conversely, replication involves devoid of programming the tasks crash to augment the system productively [6, 7].

Proper scheduling of resources is always needed [22]. Cloud has many challenges such as it is susceptible to high latency and network congestions [23]. Load balancing and service brokering are the two main topics which ensure reliability, scalability, reduced response time, increased throughput in cloud environment [24]. Designing deep learning based techniques plays a major role for human based computing in Fog and mobile edge networks [25].

Determination of the number of essential replicas and their position on the cloud is a challenging issue. Adaptive repetition of the regularly utilized data files, decisions on the number of data models, and the data nodes' positions are essential for fixing novel models based on the present cloud atmospheric situations. The repetition of data ensures accessibility and dependability, earmarking a huge model for every data, and getting a large storage space usage. A minimal number of replicas need maintenance for these files for avoiding this problem. The current cloud systems commonly employ a variety of fault detection and recovery techniques to improve system accessibility. Besides necessities currently seen, fault tolerance is a substantial property of cloud computing. It offers the assurance of appropriateness and security for the present systems in the event of a fiasco. Most of the existing works consider the system as non-faulty, but various faults may occur at every DC in real-time. To avoid such problems, there is a need to adaptively determine the number of replicas and their optimum locations, design a scheduling algorithm that should be fault-tolerant and integrate task scheduling and dynamic data replication strategies into one framework to reduce data access time and bandwidth consumption.

This paper designs a dynamic data replication framework using the Hybrid Fuzzy-CSO algorithm for the IoT-Cloud. We claim that the proposed framework is a novel model which applies an optimization algorithm for estimating the replica positions and FLD for determining the optimal number of replicas, when compared to other works on data replication in the cloud.

The main contributions of this work are.

- Developing algorithms to determine the optimal number of replicas
- Creating a Fuzzy-CSO model that provides fault tolerance, faster data access, and lower storage capacity due to replication.

The remainder of the paper is organised as follows. Section 2 presents the related works on the topic of data replication. Section 3 presents the detailed methodology of the framework. Section 4 presents experimental results and Sect. 5 presents their discussion. Finally, Sects. 6, 7 contain the conclusion and future work of the paper, respectively.

2 Related Works

Yan Wang et al. [8] have suggested a replica catalog strategy and the information retrieval technique. They have specified the deputy catalog acquisition technique for planning and duplication of the data. The nodes with the universal model of the information, replicate data sources that have huge access and elongated retort periods. Later, the Markov chain model was designed, and a matrix geometric solution was utilized. Various performance factors have been provided for enhancement of the number of models in the storage system.

DejeneBoru et al. [9] have offered models for energy drain and bandwidth requests of data access in the CC DC. They have suggested an energy effectual replication scheme based on the suggested models, which enhanced Quality of Service (QoS) with abridged communication deferments. The assessment outcomes attained with widespread imitation reveal performance and energy efficacy trade-offs and lead to the upcoming data replication resolutions strategy.

DejeneBoru et al. [10] did work on DR in CC DCs. Consequently, the abridged communication deferments and the improved QoS deliberated both energy efficacy and bandwidth depletion of the system, unlike other methods seen in the literature. The assessment outcomes got widespread replication said to reveal performance and energy efficacy rate-offs and lead the upcoming data replication resolutions strategy.

Jenn-Wei Lin et al. [11] have suggested two QoS-aware data replication procedures in the CC systems. The principal procedure agrees with the intuitive notion of high-QoS first-replication (HQFR) for making data replication. On the other hand, this covetous procedure does not have the ability to curtail data replication charge, and the amount of QoS disrupted data replicas. Using the prevailing minimum-cost maximum-flow procedure, the second procedure creates the ideal solution but consumes a large processing time than the first procedure. They also suggested node combination methods for reducing the probably huge data replication period.

Suji Gopinath et al. [12] have suggested an effective data replication scheme that vigorously repeats data based on its access reputation. This approach categorizes the data as hot, warm and cold considering the access form of the data and vigorous handling of each group replication. The replication aspect of hot and warm data is sustained based on the necessity of its accessibility.

Yalda Ebadi et al. [13] have presented a hybrid metaheuristic procedure that utilizes the universal huntability of the Particle Swarm Optimization (PSO) procedure and the indigenous hunt ability of the Tabu Search (TS) to get excellent resolutions. The outcomes indicate that the technique outperforms all of them regarding spent energy and cost.

MyunghoonJeon et al. [14] have suggested a procedure that involves variations in the user's data access form and energetically puts on an ideal replication approach. The suggested procedure has the benefit of preserving an idealist by resorting to numerous data access forms. They have done verification of the suggested procedure and legalized its efficiency.

Bo Yin et al. [15] have investigated the problem of constructing an aggregation tree for complex queries with the minimum communication cost. As complex queries have a dynamic size of intermediate results, existing Steiner tree-based approaches for traditional query operators, e.g., MIN and top-k, cannot be directly applied. They have formalized the aggregation gain by jointly considering the data pruning power (the size of data points that can be pruned during the aggregation for complex queries) and aggregation cost (the size of data points transmitted for the aggregation).

Shiming He et al. [16] have proposed a distributed joint source, routing, and channel selection scheme. The source selection issue can be concurrently solved via multipath finding. There are three sub-algorithms in this scheme, namely, interference-aware routing algorithm, channel assignment algorithm, and local routing adjustment algorithm. The interference-aware routing algorithm is used to find paths sequentially and is jointly executed with the channel assignment algorithm. After finding a new path, the local routing adjustment algorithm may be executed to locally adjust the selected paths so as to further reduce wireless interference.

Dun Cao et al. [17] have proposed a robust distance-based relay selection by optimizing the exponent-based partitioning broadcast protocol and incorporating a proposed mini-black-burst-assisted mechanism. Moreover, they developed analytic models for robust approach performances in terms of contention latency and packet delivery ratio.

ZisangXu et al. [18] have proposed a secure and computationally efficient authentication and key agreement scheme for the Internet of Vehicles (IoV), where the Road Side Units can authenticate with the vehicle. Under the analysis of the Real-Or-Random model and the simulation tool ProVerif, the proposed scheme is proved to be secure. Compared with existing schemes, the proposed scheme improves authentication efficiency and reduces energy consumption.

2.1 Research Gaps

From the above discussion of literature review, the following research gaps are identified:

- The existing approaches fail to determine the optimum number of replicas required for data replication.
- The exact locations for replica placement are randomly determined.
- The existing works assume that the system as non-faulty

Our proposed methodology fills these gaps by designing a dynamic data replication technique which determines the number of replicas and their optimum locations along with a fault-tolerant scheduling algorithm.

3 Proposed Methodology

3.1 System Model

This paper deals with data replication and task scheduling in IoT-clouds. This paper describes the device architecture that satisfies the hardware and software specifications for data collection networks and provides a prototype of its implementation on the Fuzzy-CSO algorithm's common framework for IoT-Cloud. In this paper, a Fuzzy-CSO algorithm has been proposed for designing dynamic data replication framework for IoT-Cloud network. The flow diagram of the framework is depicted in Fig. 1.

It uses the CSO algorithm to find the optimal locations for replication. The fitness function is derived in terms of the distance between the main DC and the other DCs. Then the DCs with a minimum distance to the main DC are selected as replication locations for the data. A new FLD model has been designed for estimating the required number of

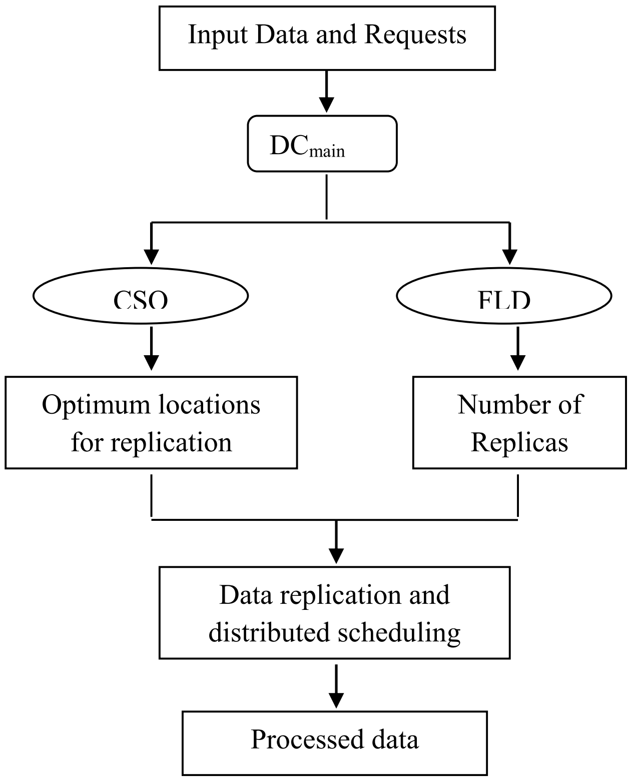
Input Data and Requests
DCmain
CSO
FLD
Optimum locations for replication
Number of Replicas
Data replication and distributed scheduling
Processed data

Fig. 1 Flow diagram of proposed Fuzz-CSO framework

replicas. The input parameters considered for the FLD model are Fault Occurrence Probability (FoP), Total Energy Cost (TEC) and the total storage capacity (TSC) of the servers. During task scheduling, data replication is performed in the selected replication points and scheduled accordingly. The uncompromised data is eventually isolated, and the entire setting is uploaded to the cloud into a predefined directory structure. This user interface allows the user the use of the request-response technique, to request for data, which can be a report of all the data, gathered that day or an average of all the data gathered that day. The system model of IoT-Cloud architecture is shown in Fig. 2. It shows a collection of sensors transmitting data to a GW, which is then responsible for uploading the together data to the DCs or Mini-Clouds.

3.2 Estimation of Energy Cost

The solution is enabled by the availability of data obtained at each node, the query workload, and the resource constraint of any node, as discussed below. The local DC and the storage IoT-clouds require internet indices for minimizing overhead responses to query data in the network sensor. Assume a cloud system containing *A DCs*. Let S_i be the storage capacity of DC *i*, $(1 \le i \le A)$. Let CD_i and CD_j be two DCs. *Let C (i,j)* and *E (i,j)* be the communication cost and energy for the storage and transmission of cloud data.

The goal of the replication strategy is to reduce the TEC and TSC of servers. Every data d_k has a principal facsimile in the main cloud DC cdb_k.

Let RP = {RP_1, RP_2… RPB} denotes replication models of all the data.

Every cdb_k has data relating to the replication model RP_k, for every data d_k.

Let $X_{i,k}$ is a binary matrix of order (A x B) such that $X_{i,k} = 1$ if the data is stored in CD_i and $X_{i,k} = 0$, otherwise.

Equation (1) states that the aggregate of the dimensions of all the data simulated at DC should not surpass S_i of cdbi.

$$\sum_{k=1}^{B} x_{i,k}\, d_k \le s_i \text{ for all } 1 \le i \le A \tag{1}$$

Let $r_{i,k}$ be the number of read operations at i passing through the cloud center.

Let CR (RP) and CW (RP) denote the cost of read and write operations under *RP*, respectively, which can be derived using Eqs. (2)–(7) [13]:

$$\text{Cost}R = \sum_{i=1}^{A} \sum_{i=1}^{B} r_{i,k}\, d_k\, C(i, dcdbi, k) \tag{2}$$

$$EnergyR = kdkE(i, dcdbi, k) \tag{3}$$

$$CR(RP) = w1.\cos tR + w2.EnergyR \tag{4}$$

$$\text{Cost}W = kdkC(i, dcdbi, k), \forall j \in wRSKj \ne iC(dcdbk, j) \tag{5}$$

$$EnergyW = kdkE(i, dcdbi, k)\forall j \in wRSKj \ne iE(cdbk, j) \tag{6}$$

$$CW(RP) = w3.\cos tW + w4.EnergyW \tag{7}$$

Every read request is retorted by $dcdb_{i,k}$ that leads to the least cost and energy.

The *TEC (RP)* for the reads and write operations is computed as

$$TEC(RP) = CR(RP) + CW(RP) \tag{8}$$

3.3 Estimation of Fault Occurrence Probability (FoP)

The probability of successful completion during the time Δt, [19] is given by

$$P(t > \Delta t) = 1 - P(t \le \Delta t) = e^{-\Delta t / m} \tag{9}$$

Let *Tc* be the calculation time for a task executed on a Virtual Machine (VM) and φ be the calculation interval between two recovery points.

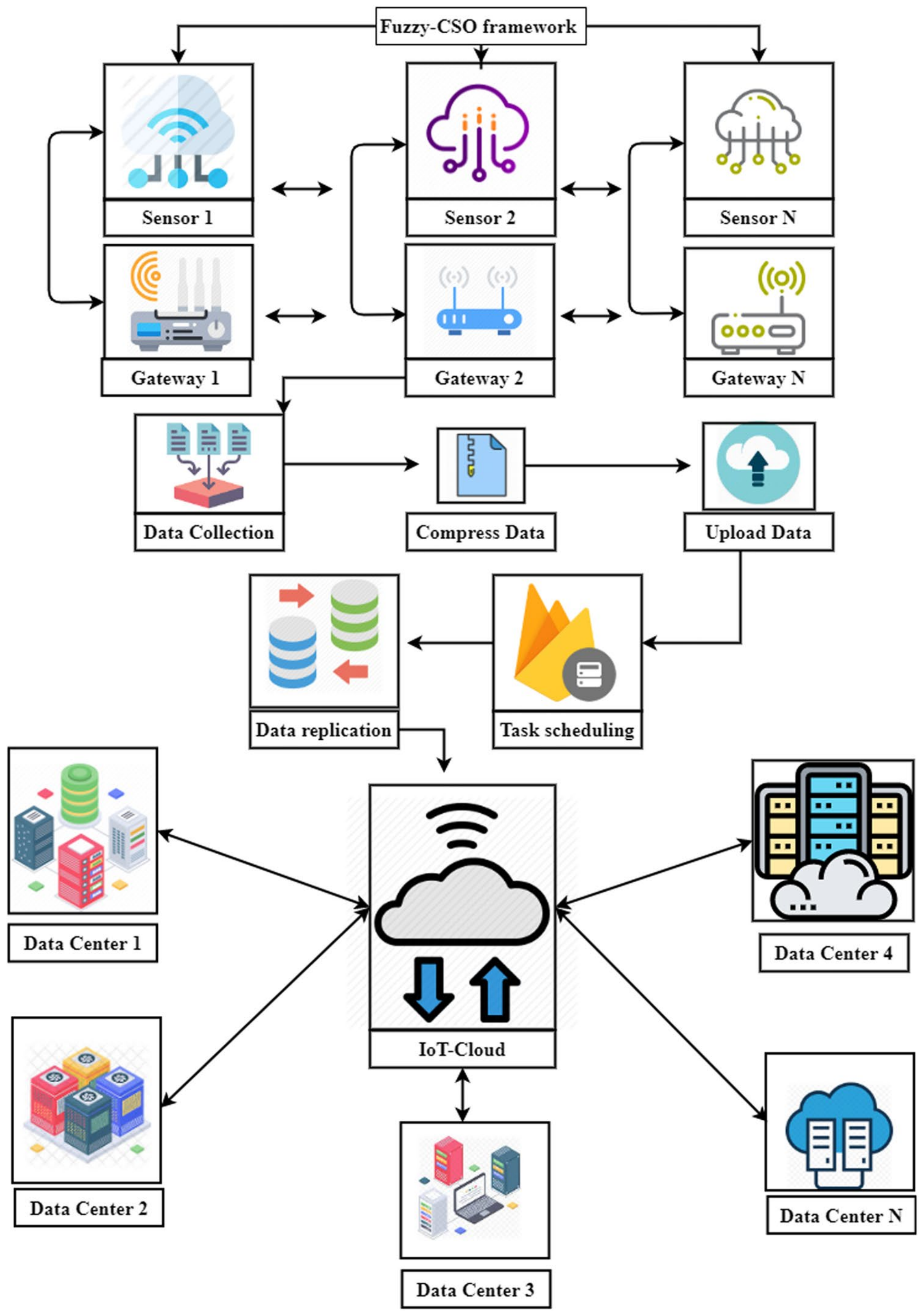

Fig. 2 IoT cloud network

Then, the mean number of attempts (NoA) required to finish Tc [20] is given by

$$NoA = \frac{T_c/\phi}{P(t > \Delta t)} = \frac{T_c\, e^{\Delta t/M}}{\phi} \tag{10}$$

The number of success (NoS) is given by

$$\mathrm{NoS} = \frac{T_c}{\phi} \tag{11}$$

The number of failures (NoF), during Δt is given by

$$\mathrm{NoF} = \mathrm{NoA} - \mathrm{NoS} \tag{12}$$

Hence

$$\text{FoP} = \text{NoF}/T_c, \\ = \frac{1}{\varphi}\left(e^{\Delta t/M} - 1\right) \tag{13}$$

3.4 CSO Algorithm

CSO is a robust and efficient method of metaheuristic swarm optimization that has obtained very positive reviews from the moment it was created. It solved several issues of optimization and added a lot of variants. The CSO algorithm was used for various test functions and focused on the normal behaviour of the cat. Compared to the PSO results obtained by the use of the algorithm with the weighting factor, the CSO algorithm has shown improved efficiency in the pursuit of the right global solutions. A number of cats are generated in CSO to start with. Each cat has its location, velocities for each dimension and fitness value. It contains a flag for the recognition of whether the cat is in a seeking mode (S_{mode}) or tracing mode (T_{mode}). The last resolution is the finest location in one of the cats as CSO retains the finest resolution till it attains the finish of reiterations [20, 21].

Table 1 shows the list of parameters used in the algorithm.

3.4.1 Seeking Mode (S_{mode})

The S_{mode} is utilized for their presentation of the cat's circumstances, which include relaxing, looking around, and seeking the succeeding location to move to. S_{mode} provides a description of four vital aspects: seeking memory pool (SMP), seeking a range of the selected dimension (SRD), counts of dimension to change (CDC), and self-position considering (SPC). SMP is utilized for a description of the dimensions of seeking memory for every cat, which specifies the cat's points. The cat may choose a point from SMP based on the instructions defined later.

The S_{mode} can be described as follows,

$$P^i = \frac{|FS_i - FS_b|}{FS_{max} - FS_{min}}, \quad where, \quad 0 < i < j \tag{14}$$

Table 1 Parameters used in algorithm

Parameter	Meaning
S_{mode}	Seeking Mode
T_{mode}	Tracing Mode
$X_{best,d}$	Location of the cat
C	Constant
R	Arbitrary Value
d_k	Data
$FS(RR)$	Fitness Function

When the objective of the fitness function is to discover the least solution, $FS_b = FS_{max}$, else $FS_b = FS_{min}$. SRD announces the mutative proportion for the chosen sizes. In S_{mode}, the variance between the novel value and the previous one will not be out of reach, as described by SRD when a measurement for a change is chosen. CDC reveals the number of sizes that are different. SPC chooses the cat's previous point, which is one of the applicants to transfer to.

3.4.2 Tracing Mode (T_{mode})

It is the sub-model for denoting the position of the cat in tracing certain objectives. Once a cat enters T_{mode}, it makes a transfer based on its velocity for each measurement.

T_{mode} can be defined as given below

$$V_{k,d} = V_{k,d} + r_1 \times c_1 \times \left(X_{best,d} - X_{k,d}\right) \quad where \quad d = 1, 2, \ldots .M \tag{15}$$

$X_{best,d}$ is the location of the cat, with the finest fitness value; $X_{k,d}$ is the location of cat k. c_1 is a constant and r_1 is an arbitrary value in the series of [0, 1].

3.4.3 Fitness Function

In finding the optimum locations of the replications, the fitness function of CSO derived is in terms of the distance between the main DC and the other DCs such that the server with minimum distance to the main DC is selected.

The fitness function FS $(RP)_i$ of a DC CD_i for replicating the data is given by

$$FS(RP)_i = D_{ds}(k, i).C\left(i, dcdb_{i,k}\right) \tag{16}$$

where D_{ds} (k,i) be the distance between cdb_k and the CD_i.

3.4.4 Process of CSO Algorithm

Mixture ratio (MR) is described by linking the S_{mode} along with T_{mode}. Perception of a cat's actions suggests that it uses this time to the fullest extent possible to be aware of resting, changing locations gingerly and slowly, and occasionally even remaining in one particular place. The S_{mode} is utilized for signifying the use of the behaviour with CSO. The deed of running after the goals of the cat is used in T_{mode}. Hence, it is evident that MR should be a small value to ensure the cats use maximum time in S_{mode}, just as that of the real world.

The procedure of CSO can be defined in 6 steps as given:

Step 1 Generation of N cats, which is utilized for updating the location.

Step 2 Arbitrarily sprinkle the cats into the M-D solution space and arbitrarily choose values close to the maximum velocity to every cat's velocities. Then randomly pick

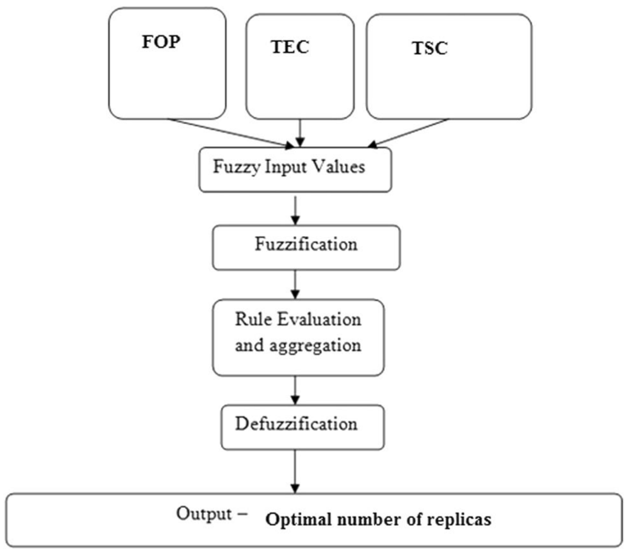
FOP
TEC
TSC
Fuzzy Input Values
Fuzzification
Rule Evaluation and aggregation
Defuzzification
Output – Optimal number of replicas

Fig. 3 System architecture of FLD

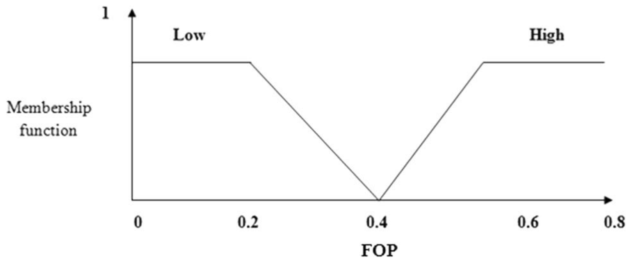

Fig. 4 MF of FOP

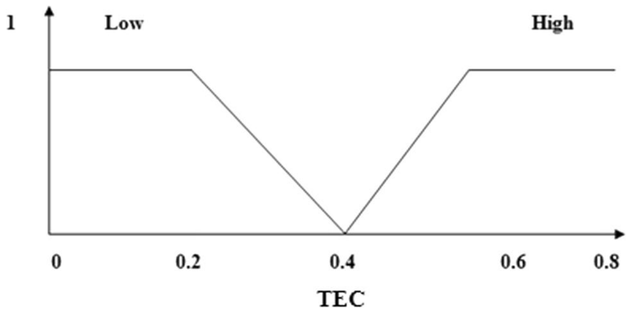

Fig. 5 MF of TEC

some cats and fix them into T_{mode} based on MR, and get the others fixed into S_{mode}.

Step 3 Calculate the fitness function FS (RP) for every replication form of data $d_{k,}$ using Eq. (16).

Step 4 Calculate the fitness value of every cat using the locations of cats, which signifies the norms of the objective, and retain the best cat in memory. A point to note is that the objective is only to recall the optimum cat's location (xbest) as it signifies the finest resolution hitherto.

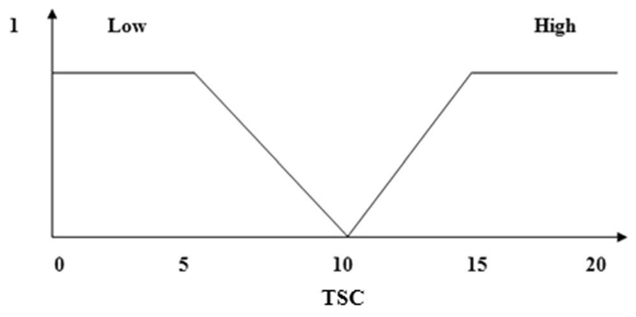

Fig. 6 MF of TSC

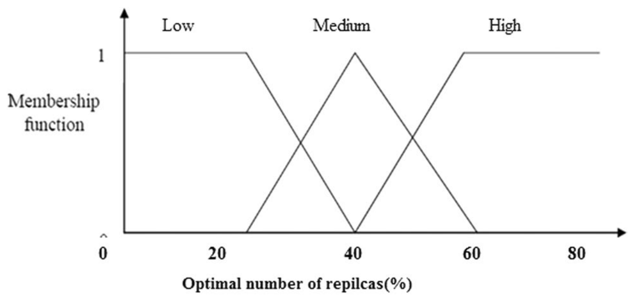

Fig. 7 MF of suspected node

Step 5 Transfer the cats based on their flags. If cat k is in S_{mode}, use the cat for the S_{mode} procedure; else use it for the T_{mode} procedure.

Step 6 Re-choose the cats and fix them into T_{mode} based on MR, and then fix the other cats into S_{mode}.

Step 7 Verify the end situation, if the required iterations are completed, dismiss the program, or else redo step 3 to step 5.

3.5 FLD Model

In CC, energy depletion and the charge of preserving novel replicas are also liable to enhance when the volume of data replicas is augmented. FLD model is used for the determination of the optimum number of replicas through consideration of the FOP, TEC and TSC metrics as input variables. It returns the optimal number of replicas from the outputs of FLD.

Figure 3 shows the system architecture of the FLD model which shows the FOP, TEC, and TSC treated as input parameters for the FLD model. The inference system consists of fuzzification, rule evaluation, and defuzzification.

Table 2 Fuzzy rules table

S.No	FOP	TEC	TSC	Optimal number of replicas (%)
1	Low	High	High	Low
2	Low	Low	High	Low
3	Low	High	Low	Low
4	Low	Low	Low	Medium
5	High	Low	Low	High
6	High	Low	High	Medium
7	High	High	Low	Medium
8	High	High	High	Low

3.5.1 Fuzzification

Figures 4, 5, 6, and 7 show the membership functions of the input and output variables. A Low Value denotes a fuzzy variable that is close to its lowest limit, and a High Value denotes a variable that is close to its upper limit. A Medium value indicates the situation in which the component is close to the average (e.g., around 0.5). Triangular membership functions are taken into account for the sake of convenience seeing their adoption in literature. Therefore, the suggested architecture is general enough, and therefore any membership functionality that best fits the application domain can be implemented. Determination of the type of Membership Function (MF) feature of primary affiliations is made since interval sets are the secondary MF.

3.5.2 Fuzzy Rules

In Table 2, fuzzy rules for the input functions and the corresponding output are shown.

As per the rules of Table 1, a maximum (High) number of replicas can be deployed when the FOP is high and TEC and TSC are low. On the other hand, when the FOP is low, there is no need for replication. Hence only a minimum (Low) number of replicas is required. A medium number of replicas are deployed in all other cases.

3.5.3 Defuzzification

It is a process in which a single number is gathered from the aggregated fuzzy set production. It is used for shifting fuzzy results into a crisp output. In other words, a decision-making algorithm that chooses the right flash value based on a flashy range is used for defuzzification. This technique defines the middle of the fused region and provides the corresponding crisp value.

In this method, a crisp value is returned from the fuzzy output set. Here, the centroid of the area scheme is considered.

Equation (17) gives a crisp value using the defuzzifier method.

$$\text{Crisp value } = \left[\sum\nolimits_{allrules} f_i * \alpha\left(\mathrm{f_i}\right)\right] / \left[\sum_{allrules} \alpha(f_i)\right] \tag{17}$$

where $\mathrm{f_i}$ refers to all rules and variables and α (f_i) is their membership function.

Algorithm 1: Determining the optimum number of replicas using the FLD model

```
Input: Measured values of DCs: FoP, TEC, TSC, and Request $R_j$
Output: Optimum number of replicas (ONrep)
For each Request$R_j$
For each DC$_i$
Find FoP (i), TEC (i) and TSC (i)
End For
Form input membership function and Fuzzy set for each variable
Form output membership function and Fuzzy set for ON$_{rep}$
Apply Fuzzy rules as per
Crisp value (HC) = Defuzzfication (Output Fuzzy set) using Eq.(19)
End For
```

The final stage is a defuzzification process in which the fuzzy outcome is converted to the level of the membership values into a single crisp value, like the fuzzification process. The defuzzification process is an inverse transition relative to the flushing process when the fuzzy output is transformed into the crisp values of the device.

3.6 Dynamic Data Replication and Scheduling

A key research challenge for investigators has been the dynamic replication technique, which can handle different changes and automatically build, remove, and maintain replicas in stochastically fluctuating distributed data storage. Therefore, only the complex approach is considered for this article. When there is not enough storage space, the algorithm specifies and determines the replica value and deletes it with the least replica life value. The following algorithm summarizes the steps involved in the dynamic replication and distributed scheduling process.

Algorithm 2: Dynamic replication and distributed scheduling

```
Input:  DCs = {$DC_1$, $DC_2$ ... $DC_n$}, Data requests from server = {$R_1$, $R_2$… $R_k$}
        Sensor devices {$S_1$, $S_2$…..$S_m$}
Output: Distributed scheduling of data from sensors
**For each** Request $R_j$, j=1, 2…k from user $U_j$
        Submit request $R_j$ to the main server $DC_{main}$
        $DC_{main}$ forwards request $R_j$ to the gateway GWv
        $GW_v$ collects data $D_j$ from the sensors $S_r$, r=1, 2…m
        $S_r$ uploads data $D_j$ to $GW_v$
        $GW_v$ transmits the collected data $D_j$ to $DC_{main}$
        $DC_{main}$ estimates the total size of data $D_{size}$
        $DC_{main}$ estimates the optimum number of replicas $ON_{rep}$ (using Algorithm-1)
        $DC_{main}$ selects the set of replicas {$DC_S$} using the CSO algorithm
        {$DC_S$} = CSO algorithm ($ON_{rep}$, $DC_1$, DC2…. $DC_n$)
        **For each** $DC_i$ Є {$DC_S$}, i=1, 2…$ON_{rep}$
                $DC_{main}$ replicates the data $D_j$ of $R_j$
        **End for**
        $DC_{main}$ schedules the request of $U_j$
        $DC_{main}$ returns the processed data to $U_j$
**End For**
```

Table 3 Experimental settings

No. of GWs	50
No. of mini-cloud storages	30
No. of uplinks or downlinks	2
GC link BW	1–5 Mbps
CC link BW	25–100 Mbps
Read delay at the gateway	25-50 MB/sec
Read delay at cloud	50–125 MB/sec
Write delay at the storage	10–25 MB/sec
Data size	100–2000 bits
No. of data items	500 to 15,000

Table 4 Data size of 64 kb and 1 Mbps bandwidth

Number of data items	Time (ms) Fuzzy-CSO	Time (ms) PSO	Time ms (ABO)
500	8250	10,625	11,644
750	9340	12,349	14,153
1000	10,225	14,627	15,181
1250	10,784	15,641	17,937
1500	12,340	17,755	19,573

In Algorithm 2, when the user requests are submitted to the main server, it forwards the respective gateway's request. The gateway then collects the data from the connected sensors and transmits them to the main server. On receipt of the gateway data, the main server estimates the total size of the data and determines the optimum number of replicas required using Algorithm 1. The optimal places or DCs for the replication of the data are selected by executing the CSO algorithm. Once the DCs are selected, the main server stores the replicated data into the corresponding DCs. It then schedules the request for execution and returns the executed results to the user.

4 Results

This section deals with the experimental setup and results for implementing the Fuzzy-CSO framework. The replication locations are determined using PSO and Artificial Butterfly Optimization (ABO) algorithms. Simulation experiments are conducted by varying the number of data items, their data Size and bandwidth. CloudSim Simulator has been used to obtain the simulation results. Table 2 shows the experimental settings.

Different data files are uploaded to the cloud storage environment, with each size in the range of [0.1, 10] GB. Each file is stored in fixed size storage unit called a block. Blocks of the same data file are scattered across different VMs. Initially, the number of replicas of each data file is 2 which are placed randomly (Table 3).

Table 5 Fixed data size of 64 kb and 8 Mbps bandwidth

Number of data items	Time (ms) Fuzzy-CSO	Time (ms) PSO	Time (ms) ABO
500	6750	7330	8850
750	7125	9527	10,127
1000	8400	10,348	12,587
1250	9600	11,599	14,392
1500	10,150	12,509	15,260

Table 6 Fixed data size of 128 kb and 1 Mbps bandwidth

Number of data items	Time (ms) Fuzzy-CSO	Time (ms) PSO	Time (ms) ABO
500	9350	11,355	12,844
750	9720	13,441	15,257
1000	10,825	15,125	17,383
1250	11,281	17,342	19,127
1500	13,150	18,925	21,540

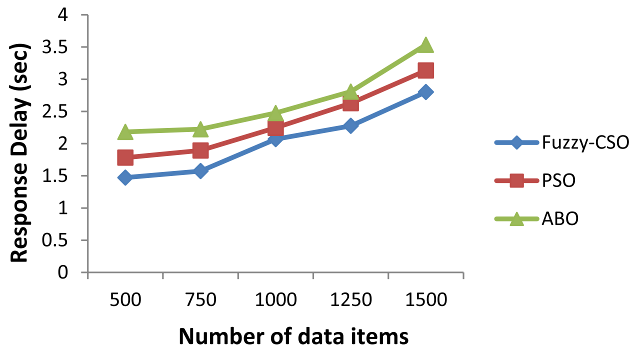

Fig. 8 Response delay for varying data items

4.1 Results of Replication Times

Tables 4, 5 and 6 show the time involved in replication of various data items to the DCs are measured for Fuzzy-CSO, PSO and ABO algorithms.

From the tables, it can be observed that the proposed Fuzzy-CSO involves lesser replication times than the other two techniques, since it adaptively determines the number of replicas.

4.2 Comparison of Performance Results

This section presents the performance corresponding to response delay, percentage of fault occurrence, bandwidth utilization (BU) and storage overhead for all the three algorithms.

Response delay The response delay R(d) is represented according to Eq. (18) as

$$R(d) = \frac{\sum_{j=1}^{Req_{ans}} W_j}{Req_{rep}} \tag{18}$$

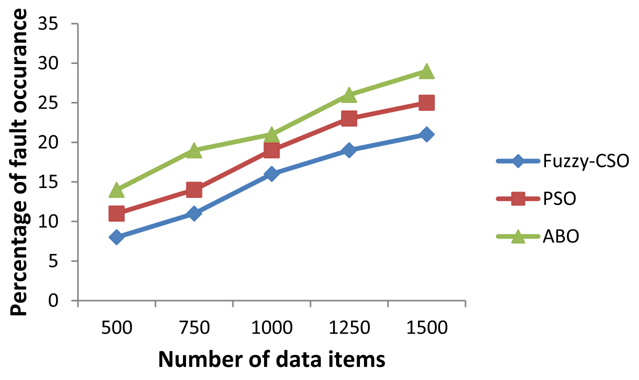

Fig. 9 Percentage of fault occurrence for varying data items

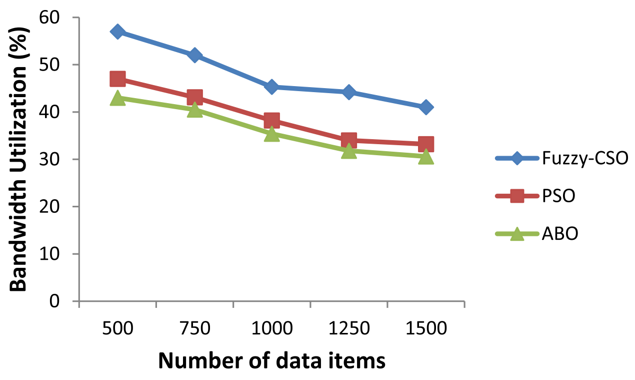

Fig. 10 Bandwidth utilization for varying data items

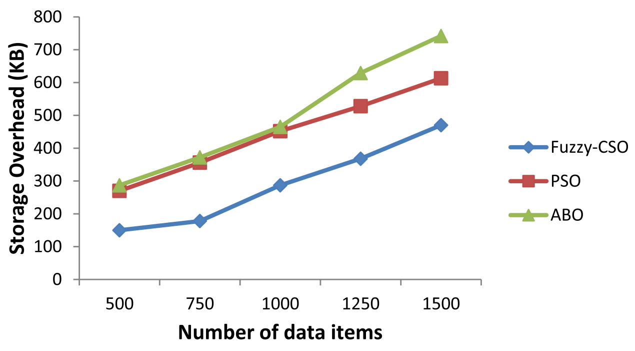

Fig. 11 Storage overhead for varying data items

where $\mathrm{Req}_{\mathrm{ans}}$ is the number of replied requests, W_{j} is the waiting time of each replied requests given by

$$\mathrm{W}_{\mathrm{j}} = \left(\mathrm{FT}_{\mathrm{j}} - \mathrm{AT}_{\mathrm{j}}\right).\ \mathrm{ST}_{\mathrm{j}} \tag{19}$$

where FT, AT and ST are the finish time, arrival time and service time.

As illustrated in the following equation, the percentage of fault occurrence is defined as

$$\%\ \text{of Fault occurrence} = (\text{No of failures/task completion time}) * 100 \tag{20}$$

BU It represents the mean percentage of the BU by calculating the average of all VMs utilization.

$$BU = \frac{\sum_{j=1}^{AVM} U(VM_j)}{AVM} \tag{21}$$

Here, AVM is the number of Assigned VMs and $\mathrm{U(VM_j)}$ is the utilization of $\mathrm{VM_j}$.

5 Discussion

Figure 8 is the graph of the response delay results for varying the data items. As derived in Eq. (18), it is obtained by summing the weighting times of all scheduled requests and then dividing it by the number of replied services. When the number of data items increases, the response delay of all the approaches increases linearly, as shown in the figure. It depicts that the delay of Fuzzy-CSO ranging from 1.4 to 2.8 s, the delay of PSO ranging from 1.7 to 3.1 s, and the response delay of ABO ranges from 2.1 to 3.5 s. Since Fuzzy-CSO determines the optimum number of replicators, unnecessary replication decisions are avoided, resulting in a reduced response delay. Thus, the response delay of Fuzzy-CSO is 13% less than PSO and 12% less than ABO.

Figure 9 is the graph showing the results of fault occurrence for varying the data items. As defined in Eq. (20), it is obtained by the ratio of number of failures occurred to the task completion times. In our work, it is assumed that the faults are single faults occurring at the physical devices. The figure depicts that the fault occurrence of Fuzzy-CSO ranging from 8 to 21, the fault occurrence of PSO ranges from 11 to 25, and the fault occurrence of ABO ranging from 14 to 29. Since the FLD model of Fuzzy-CSO considers FoP in determining the replicas, the number of fault occurrences has been reduced. Thus, Fuzzy-CSO's fault occurrence is 20% less than PSO and 33% less than ABO.

Figure 10 is the graph showing the results of bandwidth Utilization for varying the data items. As derived in Eq. (21), it is obtained by summing the bandwidth utilizations of all assigned VMs and then dividing it by the number of assigned VMs. Obviously, when more number of data items are requested, the utilization decreases linearly. The figure depicts that the bandwidth Utilization of Fuzzy-CSO ranging from 57 to 41. The bandwidth Utilization of PSO ranges from 47 to 33.2, while the bandwidth Utilization of ABO ranges from 43 to 30.6. Since Fuzzy-CSO reduces the response delay and faults, the bandwidth utilization becomes higher. Thus, the bandwidth Utilizing Fuzzy-CSO is 19% high as compared to PSO and 24% higher than ABO.

Figure 11 is the graph showing the results of storage overhead for varying the data items. It isontained by the estimating the total storage space required (in KB) for each data items. The figure depicts that the storage overhead of Fuzzy-CSO ranging from 150 to 470 and the storage overhead of PSO ranging from 270 to 613, and the storage overhead of ABO ranging from 287 to 742. Since the deployed replicas in Fuzzy-CSO reduces the energy cost and storage cost, its storage overhead is 37% less than that of PSO and 43% less than that of ABO.

6 Conclusion

The vast number of sensor GW seen across the network and sensor data from various sensors, are compressed by utilising their similarities. These compressed data are found suitable for the purpose. This article has suggested the combination of a sensor gateway, and an IoT-cloud to the sensor gateway using the FLD. It can handle any unfortunate event in the case of continuous and long-term management of the results. FLD enables easier replication and is ideal for the compression of numerical data. It is designed specifically for valued numerical sensor data and supports the effective collection of data in large volumes. This paper indicates a design for dynamic data replication and scheduling framework using the Hybrid Fuzzy-CSO algorithm for the IoT-Cloud. It uses the CSO algorithm to find the optimal locations for replications. An FLD model has been designed to determine the optimal number of replicas. During task scheduling, data replication is performed in the selected replication points and scheduled accordingly. The experimental results indicate the proposed Fuzzy-CSO framework attains minimum data transfer time, shortest response delay and higher bandwidth utilization compared to the PSO and ABO algorithms. It also has a smaller storage overhead and the least chance of developing a malfunction.

7 Future Work

The proposed technique can be useful in IoT applications like smart inpatient systems, where IoT devices and sensor nodes need to transfer a huge number of detected data regularly to the gateway in a petite period of time. Therefore, the gateway needs to validate the collaborating devices in every assembly regularly. But the main limitation of this work is how the data generated by IoT devices are transformed into information that can be used to establish a reliable and safe line of communication. Hence future work aims to enable gateway nodes, end-users and sensor nodes to authenticate each other.

Funding The authors received no specific funding for this study.

Declarations

Conflicts of interest The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Kumar A, Narendra NC and Bellur U Uploading and replicating Internet of Things (IoT) data on distributed cloud storage, In: *IEEE 9th international conference on cloud computing (CLOUD),* 2016.
2. Rao PS, Rani RU (2019) A Cuckoo search based heuristic for replicating IoT data in cloud edge system. Int J Recent Technol Eng (IJRTE) 8(5):227–235
3. Yang H and Kim Y (2019), Design and implementation of high-availability architecture for IoT-cloud services, *Sensors*, 19(15).
4. M. K. Hussein and M. H. Mousa (2014) A light-weight data replication for cloud DCs environment, *Int J Innov Res Comput Commun Eng*, 2(1).
5. Suji G, Sherly E (2018) A dynamic replica factor calculator for weighted dynamic replication management in cloud storage systems. Elsevier, Procedia Comput Sci 132:1771–1780
6. Gudadhe M and Agrawal AJ (2015) SEDReS: storage effective data replication strategy in cloud environment, Helix, *Sci Explor*, 8(6), 2018.
7. Rahmati B, Rahmani AM (2017) Data replication-based scheduling in cloud computing environment. J Adv Comput Eng Technol 3(2):297–334
8. Wang Y, Wang J (2017) An optimized replica distribution method in cloud storage system. J Control Sci Eng 2017(11):1–8
9. Boru D, Kliazovich D, Granelli F, Bouvry P and Zomaya AY, Models for efficient data replication in cloud computing Datacentres, In: *2013 IEEE globecom workshops (GC Wkshps)* , vol.18(1), pp. 446–451, 2015.
10. Boru D, Kliazovich D, Granelli F, Bouvry P, Zomaya AY (2015) Energy-efficient data replication in cloud computing Datacentres. Clust Comput 18:385–402
11. Lin JW, Chen CH, Chang JM (2013) QoS-aware data replication for data intensive applications in cloud computing systems. IEEE Transact Cloud Comput 1(1):101–115
12. Suji G, Sherly E (2017) A weighted dynamic data replication management for cloud data storage systems. Int J Appl Eng Res 12(24):15517–15524
13. Ebadi Y, and Navimipour NJ (2018) An energy-aware method for data replication in the cloud environments using a Tabu search and particle swarm optimization algorithm, *Concurr Comput Pract Experience*, 31(1)
14. Jeon M, Lim KH, Ahn H and Lee BD Dynamic data replication scheme in the cloud computing environment, In: *IEEE second symposium on network cloud computing and applications*, pp.40–47, 2012.
15. Yin B, Wei XT (2018) Communication-efficient data aggregation tree construction for complex queries in IoT applications. IEEE Internet Things J 6(2):3352–3363
16. He SM, Xie K, Xie KX, Xu C, Wang J (2019) Interference-aware multisource transmission in multiradio and multichannel wireless network. IEEE Syst J 13(3):2507–2518
17. Cao D, Zheng B, Ji B, Lei Z, Feng C (2020) A robust distance-based relay selection for message dissemination in vehicular network. Wireless Netw 26(3):1755–1771
18. Xu Z, Li X, Xu J, Liang W, Choo KKR (2021) A secure and computationally efficient authentication and key agreement scheme for internet of vehicles. Comput Electr Eng 95(C):107409
19. Wu H, Jin Q, Zhang C, Guo H (2019) A selective mirrored task based fault tolerance mechanism for big data application using cloud. Wireless Commun Mobile Comput Hindawi 2019:1–12
20. Chu SC, Tsai PW and Pan JS (2006) Cat swarm optimization, *In Proc. PRICAI 2006: Trends in Artificial Intelligence*, Guilin, China, pp. 854–858, 2006.

21. Saha SK, Ghoshal SP, Kar R, Mandal D (2013) Cat swarm optimization algorithm for optimal linear phase FIR filter design. ISA Transact Elsevier 52(6):781–794
22. Rao GS, Panem C, Anuradha T, Gad RS (2021) Emerging computational challenges in cloud computing and RTEAH algorithm based solution. J Ambient Intel Humanized Comput. https://doi.org/10.1007/s12652-021-03380-w
23. Rabie AH, Saleh AI, Ali Smart HA (2020) Electrical grids based on cloud IoT, and big data technologies: state of the art. J Ambient Intel Humanized Comput. https://doi.org/10.1007/s12652-020-02685-6
24. Jyoti A, Shrimali M, Tiwari S, Singh HS (2020) Cloud computing using load balancing and service broker policy for IT service: a taxonomy and survey. J Ambient Intell Humaniz Comput. https://doi.org/10.1007/s12652-020-01747-z
25. Gupta BB, Agrawal DP, Yamaguchi S (2018) Deep learning models for human centered computing in fog and mobile edge networks. J Ambient Intell Humaniz Comput. https://doi.org/10.1007/s12652-018-0919-8

M. Saranya did her Bachelor of Technology in Electrical and Electronics Engineering in Government College of Technology, Coimbatore. She did Her Master of Technology in Embedded Systems in College of Engineering, Anna University, Chennai. Her field of Interest is Embedded System Design. Currently, she is pursuing her Research in College of Engineering, Anna University, Chennai.

R. Ramesh did his Bachelors in Engineering in Electrical and Electronics Engineering in Hindustan College of Engineering and Master's degree in Power System Engineering at Annamalai University. He obtained his Ph.D in the field of Power Systems from College of Engineering, Anna University, Chennai. Currently, he is working as Professor in the Department of Electrical and Electronics Engineering in College of Engineering, Anna University, Chennai. His areas of interest are Embedded System, Wireless Communication Technology and Smart grid.