A Highly Functional Ensemble of Improved Chaos Sparrow Search Optimization Algorithm and Enhanced Sun Flower Optimization Algorithm for Query Optimization in Big Data

Mursubai Sandhya Rani*, Dr. N. Raghavendra Sai
Department of Computer Science and Engineering, Koneru Lakshmaiah Educational Foundation,
Vaddeswaram, Andhra Pradesh, India.

Abstract—**Numerous systems have to provide the highest level of performance feasible to their users due to the present accessibility of enormous datasets and scalability needs. Efficiency in big data is measurable in terms of the speed at which queries are executed physically. It is too demanding on big data for queries to be executed on time to satisfy users' needs. The query optimizer, one of the critical parts of big data that selects the best query execution plan and subsequently influences the query execution duration, is the primary focus of this research. Therefore, a well-designed query enables the user to obtain results in the required time and enhances the credibility of the associated application. This research suggested an enhanced query optimizing method for big data (BD) utilizing the ICSSOA-ESFOA algorithm (Improved Chaos Sparrow Search Optimization Algorithm- Enhanced Sun Flower Optimization algorithm) with HDFS Map Reduce to avoid the challenges associated with the optimization of queries. The essential features are extracted by employing the ResNet50V2 approach. Effective data arrangement is necessary for making sense of large and complex datasets. For this purpose, we ensemble Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Improved Spectral Clustering (ISC). The experimental findings demonstrate a significant benefit of the proposed strategy over the present optimization of the queries paradigm, and the proposed approach obtains less execution time and memory consumption. The experimental results show that the proposed strategy significantly outperforms the current optimization paradigm, reaching 99.5% accuracy, 29.4 seconds of execution time, and 450 MB less memory use.**

Keywords—Big data (BD); query optimization; Improved Chaos Sparrow Search Optimization Algorithm (ICSSOA); Enhanced Sun Flower Optimization Algorithm (ESOA); ResNet50V2; DBSCAN

I. Introduction

Big data empowers businesses to make informed decisions and take appropriate action by allowing them to examine enormous data in volume, variety, and velocity [1]. Big data can be stored and queried using a variety of databases and data structures: Relational databases are employed for read-intensive analytic queries; Internet transaction processor platforms are utilized for faster uploads and reliability; NoSQL storage systems are used for handling massive volumes of data [2, 3]. Different data stores have been created and constructed for various purposes and the best results. SQL databases are effective at storing and processing structured data, but their efficiency suffers from read-intensive queries. Similarly to how NoSQL storage systems are tailored to deal with unstructured data, columnar databases are utilized for the analytic processing of queries [4-6].

The information that has been processed is kept in several databases so that analysts can use it. Performance optimization and various data structures are crucial for applications that use a lot of data [7, 8]. Building scalable and effective data pipelines is a significant difficulty. These data pipelines, which are vital to the functionality of the applications, are optimized and maintained by data engineers [9]. Researchers and data scientists utilize the data warehouse to analyze, evolve, and load the data for their research projects. The enhancement of query efficiency and extra complexity brought on by the various data models employed in these databases present ongoing challenges for big data platforms that use these databases [10-12].

The many Operation SITE Allocation (OSA) strategies to execute the query are born from the advancement of query optimization. OSA problems are sought after to improve query execution plans in terms of system throughput or response times [13]. The query optimizer's three main parts are "Cost Model," "Search Space," and "Search Strategy." Designing the various cost coefficients and the objective function is the responsibility of the cost model. A variety of different query execution strategies are represented by the search space [14, 15]. The search method is also used to probe the search space to find the most promising query execution technique.

Previously, deterministic optimization methods and a variety of databases were used for query optimization. Only basic CDSS queries are a good fit for deterministic algorithms [16-18]. Nature Inspired Computing (NIC) has tremendous prospects for computational intelligence and is now being applied to address CDSS query optimization concerns. There is a long list of NIC computing techniques, some of which depend on the genetics of animals, insects, birds, and people, as well as on music and water [19]. The most admired NICs include Artificial Bee Colony, Cuckoo Search, Ant Colony Optimization, Grey Wolf Algorithm, and Genetic Algorithm. After reviewing the literature on query optimization, it was discovered that distributed CDSS queries had received a lack of attention. To speed up the data retrieval, a creative query optimizer is required. The suggested query optimizer helps identify an ideal query execution plan that reduces the overall consumption of I/O, computing, and communication resources [20].

The increasing scale and complexity of big data have made query optimization a critical challenge. Existing methods often struggle with several limitations, including high computational cost, slow convergence, and inefficiency when handling large, distributed datasets. Many traditional techniques are also unable to address data skew effectively, ensure quick response times, or optimize query execution under heavy query loads. These shortcomings highlight the need for a more efficient approach to query optimization that can scale with growing data volumes and provide faster, more resource-efficient execution in modern big data environments. To address these challenges, we propose an enhanced query optimization method that significantly improves execution time and reduces memory consumption, making it better suited for the demands of today's data-driven applications.

To tackle the issue mentioned above, we introduced a novel approach to big data arrangement and feature extraction. This reduces the execution time, retrieval time, and memory usage. Compared with existing methods, the proposed approach performs better.

A. *Research Contribution*

The key objectives of this research are as follows:

- Initially, we employed a secure hash algorithm in preprocessing to find the hash value. Then, centered on the HV, the map reduction process is executed.
- After the removal of repeated data, the essential features are extracted by employing ResNet50V2.
- Entropy values are inputted to the deep adaptive hybrid clustering algorithm DBSCAN and spectral clustering for the big data arrangement.
- Finally, the query is optimized with the help of the ensemble Improved Chaos Sparrow Search Optimization algorithm (ICSSOA) and Enhanced Sun Flower Optimization algorithm (ESFOA).

The following part of the article is structured as follows. The existing prior works are briefly described in Section II. The proposed strategy is described in detail in Section III. The suggested method is extensively simulated in Section IV. Section V provides the conclusion.

II. Related Works

Some existing prior works related to significant data query optimization are analyzed in this section.

An improved query optimizer known as CDSS was modelled by Sharma et al. [21] using a hybridization firefly-genetic algorithm (GA) on a constrained divergence environment (RDFG_CDQO). This CDSS was created with the goal of achieving the best query execution plan possible to reduce processing, input-output, and interaction demands when running CDSS queries. The controlled GA's slower convergence difficulty would be cautiously defeated by the enhanced utilization of the CDSS technique, achieving significant variance in "2" successive generations. The CDSS optimizer could not solve the QO issues. For the query retrieving rate, Lekshmi et al. [22] presented the Top-k Query Multi-Keyword Threshold method (Top-k QMKST). The query and many keywords are primarily divided, and B+ tree indexing was used to execute the data index. Response time and spatial complexity were both decreased by employing Top-k QMKST. The Kullback Leibler Divergence also uses the index list of terms to determine a score value. The results of the experimental study show that the suggested technique performs better.

For the skewed-ranging queries, Wei Ge et al. [23] suggested a method known as correlation-aware partitions. In the form of a geometrical curve-fitting problem, it introduced a problem known as partitioning optimization on continuously correlated data. The boundaries of the range query must be used to partition data optimally. The boundary for the range was utilized in this case to incorporate the best partitions and significantly reduce the computational cost compared to the standard dynamic programming. When compared to the global one, the local one performed better instead of attempting to increase effectiveness.

Sinha et al. [24] proposed an approach for distributed datasets by combining the genetic algorithm (GA) and the k-means clustering method. The suggested strategy is divided into two phases; in the initial stage, parallel GA is performed to data chunks spread across many machines. GA takes into account the covariance among the data sets and offers an improved summary of the original information. Phase 2 applies K-means with K-means++ initialization on the intermediate output to produce the outcome.

Ansari et al. [25] suggested a parallel variant of the conventional K-means algorithm for use in the Hadoop distributed environment. The results of the experiments demonstrate that the suggested K-means algorithm operates better than conventional K-means when clustering a significant volume of datasets. Compared to current methods, the suggested approach produces better results.

A. *Research Gap*

Existing query optimization techniques, including Top-k QMKST (Lekshmi et al. [22]) and the CDSS optimizer (Sharma et al. [21]), concentrate on increasing query execution efficiency but struggle to handle dynamic or large-scale datasets. Top-k QMKST speeds up response times but might not be able to handle high-dimensional data effectively, and the CDSS optimizer enhances convergence but has trouble optimizing query retrieval rates. Other methods that deal with partitioning and data summarization, including correlation-aware partitions (Wei Ge et al. [23]) and the integration of evolutionary algorithms with K-means clustering (Sinha et al. [24]), do not sufficiently improve query execution in distributed systems with big datasets. Furthermore, the parallel K-means approach of Ansari et al. [25] enhances clustering but ignores memory usage and query execution time. By using ResNet50V2 for feature extraction, the ICSSOA-ESFOA method for improved query optimization, and DBSCAN and ISC in combination for efficient data arrangement, our proposed work seeks to close these gaps. By addressing the shortcomings of current techniques, our strategy guarantees quicker query execution, better memory management, and increased scalability in significant data contexts.

III. Proposed Methodology

In order to handle and store BD, which is extremely large in volume and contains numerous data models, organizations maintain various databases. For business purposes, it is essential to query and analyze BD for insight. In this study, the ICSSOA-ESOA algorithm and the HDFS map-reduce approach were used to improve the query optimizer procedure in BD.

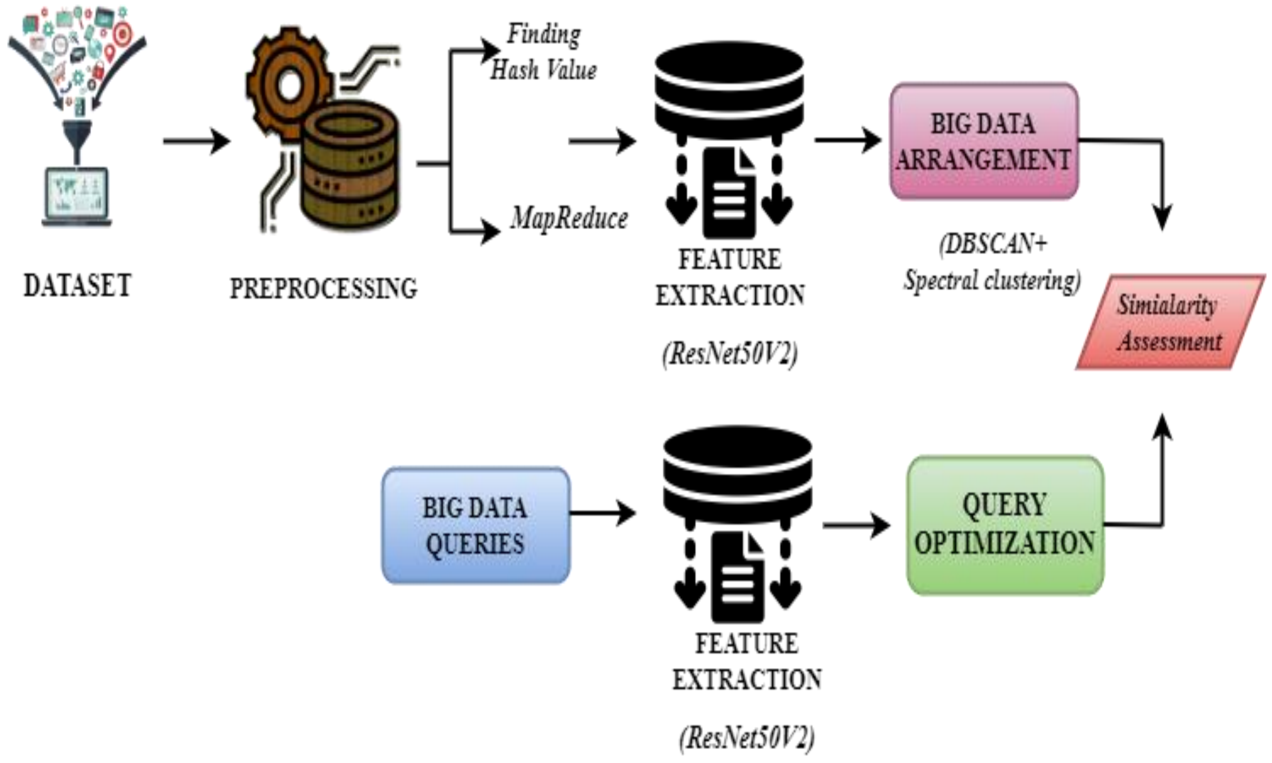


Fig. 1. Proposed methodology architecture diagram.

To extract the essential features from a big dataset, we employed ResNet50V2. Then, the big data are arranged with the help of an ensemble DBSCAN approach and an improved spectral clustering approach. The proposed approach is analyzed and evaluated by using four benchmark datasets. The overall framework of the proposed approach is shown in Fig. 1.

A. *Problem Statement*

The number of datasets that need to be evaluated is increasing, necessitating several databases to store the preprocessed data in various information formats. Several methods, like materialized views and data cubes, can decrease query latency but necessitate significant computation and preparation. In order to deliver estimated results with error bounds, approximate query processing (AQP) was implemented. Nowadays, the majority of AQP models only support one database. The suggested AQP model supports heterogeneous databases with various data models by keeping up-to-date samples in a single database. Any database can be used to conduct the SQL query. The query optimizer chooses the samples automatically and provides users with approximations of the results. For this purpose, we introduced a novel approach for query optimization.

B. *Preprocessing*

The pre-processing of the input data was carried out during this phase. First, it uses the Secure Hash Algorithm (SHA-3) to determine the HV for every bit of data. Then, using HDFS, the MR process is carried out using the HV as its focal point. The subsections below explain the SHA-3 and HDFS processes. The SHA-3 algorithm is specified for a digest length d with a value of 224, 256, 384, or 512 and a message M with two bits "01" inserted at the conclusion, such that$SHA-d(M) = KECCAK(c)(M||01, d)$, while SHA3 and KECCAK are functions, M is the input string to the SHA-3 method.

1) The SHA-3 algorithm is utilized to find the hash value of big data: Utilizing permutation functions, the SHA-3 method, also referred to as the Keccak algorithm, was created. Keccak performs encryption well and has a high degree of attack resistance. SHA-3 is safer than earlier iterations like SHA-1 and SHA-2. The SHA-3 method can provide multiple fixed-bit hash values for different input bits. The outcome of this research is a 256-bit hash value.

2) Map and reduce: The two most crucial MapReduce processes are the "Map and Reduce" operations. The Apache

Foundation created the distributed system infrastructure known as Hadoop. Users can fully leverage the platform's massive data storage and quick computation capabilities by developing distributed applications without familiarity with the architecture's inner workings. Hadoop implements a distributed file system called HDFS. Although HDFS requires the usage of costly hardware, it provides good features and strong fault tolerance. Additionally, it offers a fast interface for accessing application data, making it appropriate for programs with big data sets. HDFS lowers the file system's restrictions for accessing the data in stream form. HDFS and Map Reduce are the two main components of the Hadoop system. Massive data storage is primarily provided by HDFS, and distributed computing functions are supplied by Map Reduce. The simple description of Hadoop's data processing is that the Hadoop cluster analyzed the data to produce its outcomes. In Fig. 2, the method of processing flow is depicted.

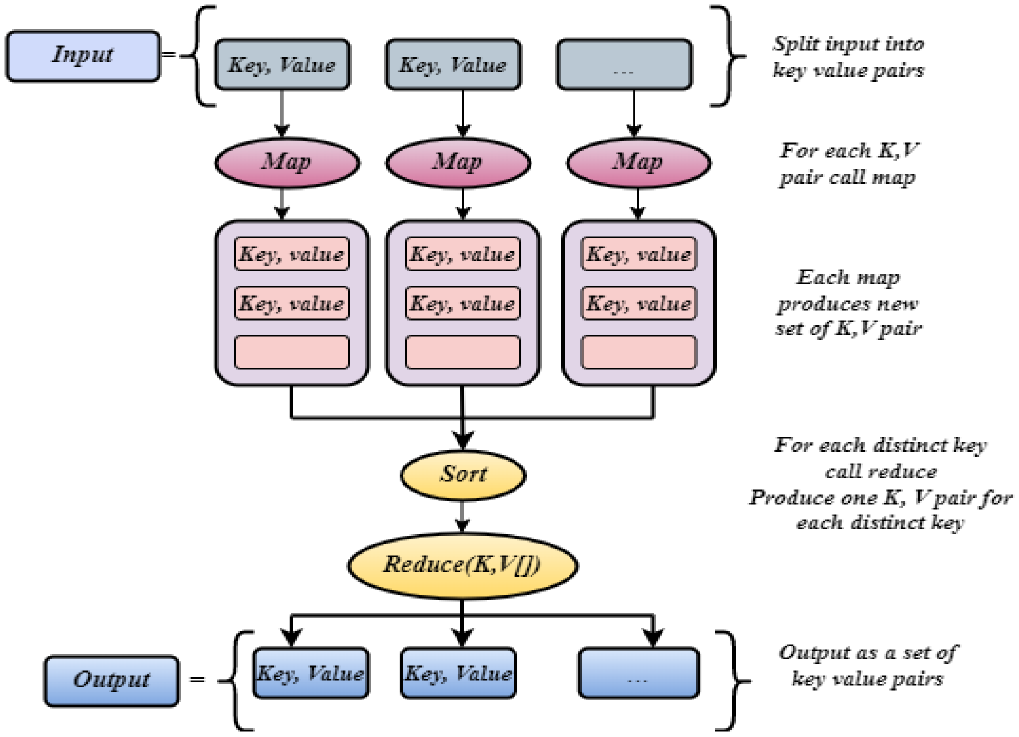

Input
=
Key, Value
Key, Value
...
Split input into key value pairs
Map
Map
Map
For each K,V pair call map
Key, value
Key, value
Each map produces new set of K,V pair
Sort
For each distinct key call reduce
Produce one K, V pair for each distinct key
Reduce(K,V[])
Output
=
Key, Value
Key, Value
...
Output as a set of key value pairs

Fig. 2. Framework of map and reduce.

HDFS and MapReduce are the two main parts of Hadoop, as shown in Fig. 3. The storage of enormous amounts of data is the responsibility of HDFS, and the processing of massive amounts of data is a function of MapReduce. Another two crucial parts of Hadoop are the distributed database system Hbase and the data warehouse tool Hive. Records are kept in a Hadoop cluster using the HDFS. The HDFS interface resembles a straightforward hierarchical file system with straightforward operations like adding, deleting, moving, and more. However, the HDFS files are broken up into data blocks based on specific requirements, and then a massive number of data blocks are distributed over numerous slave nodes. It departs significantly from conventional storage structures at this point. The user typically chooses the number of data blocks to put and the dimension of each separated data block.

MapReduce, which includes Job Trackers and Task Trackers, is DFS's top layer. Massive files are partitioned into equal sections by default on HDFS. This default value is set at 64 M in the HDFS overview document. The data file 1 has been separated into three portions and placed in three distinct machines. Map Reduce is a task that is called Map and computes after every Hadoop input component. The system will move through each input data individually in the task before analyzing the map and turning it into a key-value format. The outcome will be produced in the key-value pair's form. As an input to Reduce by key, Hadoop will then transmit the outcome of the preceding phase. The Reduce Task's results, retained on HDFS, are the outcome of the entire task.

C. Feature Extraction

Datasets in big data scenarios may contain a large number of variables or attributes and be exceedingly high dimensional. High dimensionality can present difficulties in overfitting, poor interpretability, and computation complexity. Feature extraction algorithms can reduce dimensionality by converting the original features into a lower-dimensional representation while maintaining the crucial data. Analysis and modelling could become more effective as a result. From the original data, the significant aspects are retrieved, including closed frequent item set, support, and confidence. Finally, entropy computation is used to regulate confidence and support value. The following part provides an overview of the extraction of feature processes.

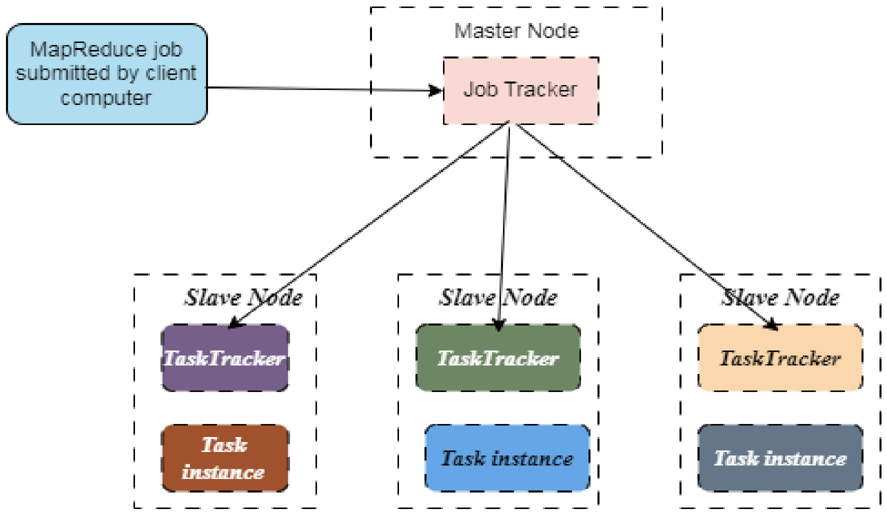


Fig. 3. Hadoop's two core components.

1) ResNet50V2: Deep feature extraction is illustrated in this subsection. Deep feature extraction employs deep neural networks to extract significant and valuable information from raw data. These characteristics capture high-level representations that are more useful for handling the current task. For query optimization in big data, we used the ResNet50V2 framework as a deep extraction of features method. ResNet50V2 represents a convolutional neural network (CNN) that excels in various computer vision applications. To tackle the degradation issue in deep networks, a variation of the ResNet design is used, which uses skip connections.

The 50-layer ResNet50V2 was pre-trained using a sizable dataset, such as a big datasets. The network can learn residual mappings through the use of residual blocks, which also makes it easier to train deeper networks. The skip connections also facilitate the direct transfer of gradients from the initial layers to subsequent layers, which improves training. Due to its ability to extract complicated and structured patterns from big data, the ResNet50V2 architecture is advantageous for query optimization feature extraction.

The deep layers of ResNet50V2 enable it to learn abstract representations. The benefit of Transfer Learning may be obtained by utilizing the pre-trained ResNet50V2 approach, as it has previously acquired general features from a sizable dataset like the hospital compare, Twitter, and IMDb datasets. ResNet50V2 can record generalized representations tuned for query optimization owing to this pre-training. The precision and effectiveness of the query optimization can be improved by applying the learned features from ResNet50V2.

The ResNet50V2 features provide a more advanced representation of the input optimization of queries, capturing essential data for positions, including bid arrangement of data. We may utilize the potent representations learned by ResNet50V2 by using these features as inputs for multiple machine learning algorithms. By doing that, we want to improve the precision and functionality of our query optimization mechanism. ResNet50V2's high-level features enable a more thorough and insightful representation of the input data, enhancing our capacity and eventually enabling improved optimization.

D. Big Data Arrangement

Big data arrangement is a key component of the data management process, which involves structuring and organizing enormous amounts of data to facilitate effective analysis, storage, and retrieval. For clustering and pattern recognition tasks in data analysis and deep learning, ensemble DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and Improved Spectral Clustering can be particularly beneficial. Combining DBSCAN with Spectral Clustering can take advantage of each technique's advantages as each approach has advantages and disadvantages of its own. The proposed method achieves improved noise handling, improved cluster separation, scalability, merging local and global information, handling variable cluster Densities, and more while combining the methodologies.

1) DBSCAN clustering algorithm: DBSCAN, a popular density-based clustering technique, can locate several clusters based on the predicted density distribution. It can detect shaped clusters and does not require prior knowledge of the cluster size. The following examples show the core concept of DBSCAN. DBSCAN collects all points in the neighbourhood of a random, unvisited point called p, while p is the initial location and r is the neighbourhood's maximal radius. The minimal number of units needed to generate a dense zone is called the density threshold MinPts. If MinPts points or more are nearby, point p is a core point. All of the points in p, ϵ-neighbourhood are put into an identical cluster if p is the centre point together with all of the other points in p. DBSCAN locates all density-reachable points. It includes them in the same cluster for every point in the cluster. If point q is densely accessible from other core points but has a smaller neighbourhood than MinPts, it is also a border point that belongs to the cluster. An isolated or noisy point cannot be reached from any other point. Using consecutive cluster extraction, DBSCAN completes the clustering procedure. A finalized cluster is created by iterating this procedure till no more density-reachable spots are discovered. The three categories that DBSCAN uses to categorize a set of points are noise, low-density boundary points, and high-density core points. The following are three different types of points' definitions.

2) Initialization of the variables: In K-DBSCAN, the HS is optimized to get the best clustering parameters. Thus, "Eps" and "Minpts," the two clustering parameters for input, have been utilized as the HS's decision variables, correspondingly. Given that the set of data is split into categories that are considered as K, every parameter variable's maximum value shouldn't be greater than the K-equal partitions of the entire data set. These two variables are initialized with the following values:

$$Eps \in \left(0, \frac{SDR}{2 \times K}\right) \quad (1)$$

$$Minpts \in \left[1, \frac{Num\_obj}{\left(\frac{LDR}{SDR}\right) \times K \times D}\right] \quad (2)$$

While SDR and LDR are the smallest value and greatest values across all dimension that ranges from the entire data set, accordingly, the variable shows the number of objects utilized for clusteringNum_obj. The dimension is denoted by D.

3) The objective function: A multi-objective collaborative evaluation approach is provided for the HS in the K-DBSCAN optimization issue. The overall number of clusters produced by DBSCAN under different parameter variables is monitored by using the initial target function, which can be shown as the total amount of variance among that and the determined clustering number K. Since the main objective of this clustering approach is to produce K groups, this variance can be expressed as the total amount of variance between it and the established clustering number K.

$$Minimize f1 = |c - K| \quad (3)$$

The total number of clusters is represented as K, which has been predetermined, and the real number of clusters is indicated as c DBSCAN, which has been produced using the current set of decision variables.

The DBSCAN method can identify unusual noise. When the outcomes of the parameters "Eps" and "Minpts" are improperly chosen, particularly if they are disproportionately matched, it may result in under-differentiation, where most or even all of the data items are misidentified for outliers.

Two distinct groups make up the initial data set in Fig. 4, and Fig. 5 displays the results of clustering with excessive noise caused by subpar clustering parameters. Acquiring the cluster number of 2 is possible, although many valid points are confused for noise entities. Consequently, a separate function of the multi-objective optimization method is utilized to maximize the number of objects in the least efficient cluster and prevent such an abnormal occurrence.

$$Maximize f2 = num(s\_clusters) \quad (4)$$

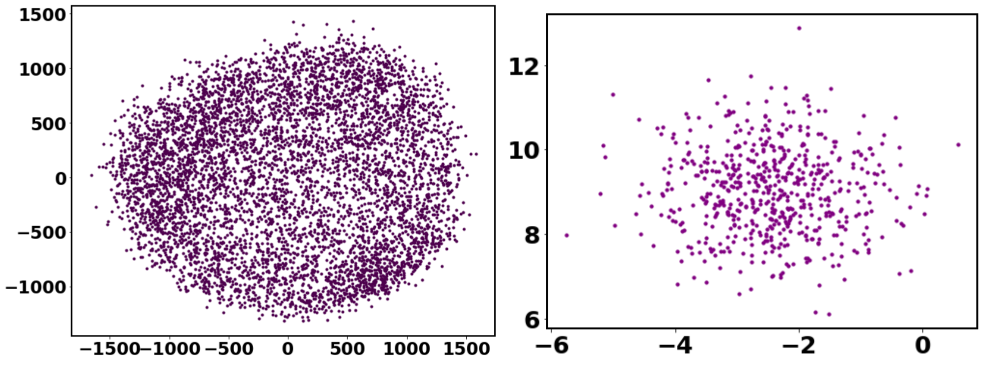


Fig. 4. The initial formation of the dataset.

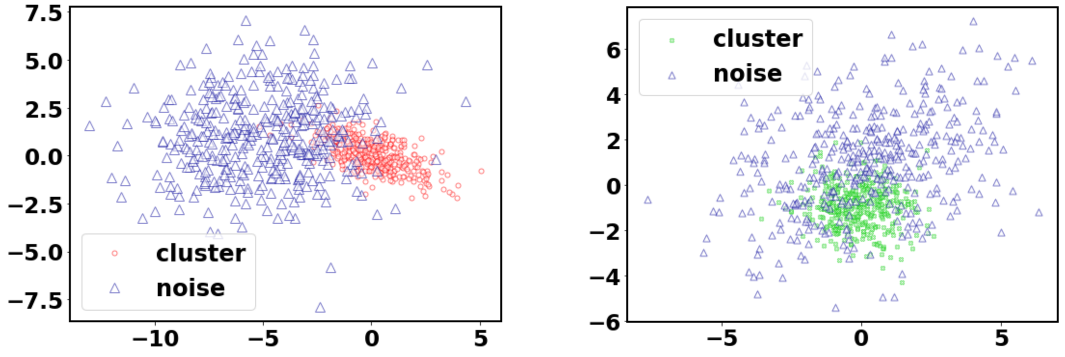


Fig. 5. Noise in clusters.

The term $num(s\_clusters)$ refers to the number of items in the smallest practical cluster. Consequently, the following is an expression for the K-DBSCAN's multi-objective collaborative evaluation function:

$$F = (Minimize\ \ f1,\ Maximize\ \ f2) \tag{5}$$

Obtaining the necessary K clusters is the primary objective of K-DBSCAN. This is followed by the effect of clustering that produces the fewest inaccurate noise objects. In other words, $f1$it has a greater priority than$f2$, which is indicated by the notation:$f1 \lhd f2$.

4) Framework: According to the information above, the two clustering factors, "Eps" and "Minpts," are used in DBSCAN as the HS variables for decision-making. The multi-objective collaborative evaluation function can be used with the clustering parameter's optimal value to get a superior clustering outcome with K categorization when using DBSCAN.

Additionally, relatively low parameter values typically result in a superior clustering effect when using the DBSCAN algorithm. The size of "Minpts" indicates a significant impact on how well noise of clustering is judged under the condition of a specific parameter "Eps," and the larger it is, the more probable it is that genuine data will be viewed as noise objects. Thus, the variable of decision "Minpts" has been set to a number that enhances over time with the repetition stage process to acquire adequate clustering factors, including.

$$Minpts = Minptsminmax_{min} \tag{6}$$

While gnit denotes the number for the generation currently in use, NI is the maximum number of repetitions and $Minpts_{max}Minpts_{min}$denotes the variable upper and lower bounds, accordingly.

5) Spectral clustering: Typical graph-based clustering techniques include Spectral Clustering without monitoring the data. Techniques for Spectral Clustering often start with local data that has been encoded in a weighted network of information and then aggregates according to the associated similarity matrix's global characteristic vectors. In Spectral Clustering, a function of mapping that explicitly maps characteristics to the group tag matrix is automatically learned for every task to anticipate cluster tags.

The process of learning can automatically use dissimilar data to enhance clustering efficiency. In Spectral Clustering, communities of nodes connected near one another are characterized in a graph using a method known as clustering. The nodes are placed in a low-dimensional area that can be easily segmented into clusters. Affinity, Degree, and Laplacian matrices and other specific values of these matrices produced from a graph or data collection are used in spectral clustering. The crucial steps in creating a Spectral Clustering algorithm are as follows:

Prior to using the spectral clustering procedure, we must first Figure out the matrix for similarity, which is then indicated as the overlap matrix of degree P. It can be shown as,

$$p = \begin{bmatrix} 0 & p_{1,2} & \dots & \cdots p_{1,n} \\ p_{2,1} & 0 & \dots & \cdots p_{2,n} \\ \vdots & \cdots & 0 & \vdots\ \vdots \\ p_{n,1} & \cdots & p_{n,n-1} & 0\ \ \vdots \end{bmatrix} \tag{7}$$

For the arrangement criterion, we may assume that every request is split into k1, k2, and two groups; this work employs the conventional division approach. Suppose q is a vector. These are the definitions of the qi elements:

$$q_i = \begin{cases} \sqrt{\frac{d_2}{d_1 d}} & , \mathrm{i} \in k_1 \\ -\sqrt{\frac{d_1}{d_2 d}} & , \mathrm{i} \in k_2 \end{cases} \tag{8}$$

In the event that the cluster indicator matrix $F \in R^{n\times k}$is correct. Assuming consistent with each perspective, we can define the clustering of spectral data issues as,

$$\min_{F,F^TF=1} \sum_{v=1}^{t} Tr\ (F^T L^v F) \tag{9}$$

While every graph evenly contributes to the outcome F. We ignore the specifics of the graph creation in the equation above. Several additional studies just take the mean of the vertices and then implement the spectral clustering independently instead of mandating that multiple graphs share the same F.

Improved Spectral Clustering Algorithm (ISCM). We provide an improved spectral clustering technique (ISCM) relying on the enhanced k-means algorithm. The approach accomplishes secondary clustering in addition to resolving the initial value issue. We take into account the parameters as previously mentioned in accordance with the QoS criterion. We may determine whether secondary clustering is necessary by evaluating the variable sizes before the method operates. There is no need to recluster if the present QoS exceeds the users' desire to allocate resources once the strategy has been performed. The clustering spectral optimization scheduling algorithm's implementation procedures are then described.

E. Query optimization

Big data systems frequently handle enormous amounts of data. By dramatically reducing the time it takes for a query to execute, query optimization can guarantee that users or applications can quickly and effectively retrieve the needed data. Query optimization aids in efficient resource allocation, cutting costs and guaranteeing the best use of available resources. It minimizes hardware waste and prevents nodes from becoming overloaded. For this purpose, we ensemble the Improved Chaos Sparrow Search algorithm (ICSSA) and Enhanced Sun flower optimization algorithm (ESFO). ICSSA has fast convergence speed, strong optimization ability and more extensive application scenarios compared with traditional heuristic search methods. Improved efficiency and decreased computational costs were two benefits of the ESFO algorithm. We ensemble both algorithm's merits to effectively optimize the query.

1) Sparrow search algorithm: The SSA bases its description of the sparrows' predatory and anti-predatory behavior for updated locations on the following guiding concepts. The population of sparrows is split into followers and producers. The sparrow's two identities may be switched around, and everyone has a system for detecting danger. Every sparrow, in particular, is sensitive to potential threats or natural enemies and will immediately begin anti-predatory activity to defend itself. The producers are highly active, adept at foraging for food, travel widely, and lead other sparrows on their quest. To increase their food intake by snatching it or foraging nearby, seekers seek the producer and follow them to find additional food.

2) Basic concepts: The individual matrix is displayed below, with N sparrows assumed to be in D-dimensional space.

$$X = [x_1, x_2, .. x_N]^T, x_i = [x_{i,1}, x_{i,2}, \dots, x_{i,D}] \quad (10)$$

While xi, D denotes the ith sparrow's location in the D dimension.

$$x_{i,j}^{t+1} = \begin{cases} x_{i,j}^t \bullet exp\left(\frac{-i}{\alpha \bullet iter_{max}}()_2\right) \\ x_{i,j}^t + Q \bullet L \quad R_2 \geq ST \end{cases} \{ \quad (11)$$

The present iteration count, t, is represented here. Itermax indicates the greatest amount of the iterations$j = 1,2, \dots, d$. It falls between 0 to 1 and is a uniform randomized value. The warning and security values for sparrows are represented by $R_2 (R_2 \in (0,1))$ and$ST (ST \in (0.5,1.0))$. An ordinary distribution characterizes a random number Q. Every matrix's 1d elements comprise the L matrix. When this $R_2 < ST$occurs, the provider adopts a wide-area search phase while they are not in danger from any natural competitors and are in a generally safe environment. Due to Eq. (3), the follower position is upgraded.

$$x_{i,j}^{t+1} = \begin{cases} Q.exp\left(\frac{x_{worst}^t - x_{i,j}^t}{i^2}\right) & \text{i} > \frac{n}{2} \\ x_p^{t+1} + |x_{i,j}^t - x_p^{t+1}| \bullet A^+ \bullet L & \text{otherwise} \end{cases} \quad (12)$$

In which x^t worst indicates the current position of the bird with the worst adaptability. The spot of the bird with the best producer adaption is represented by the numberx_p. Every component of the matrix shown by A is represented by a value at random of one or zero. A+ equals$A^T AA^{T-1}$.

3) Danger awareness mechanism: When hunting, sparrows sense the danger of hunt and may fly away from their current location and to another. The individual sparrows that detect danger often range between 10% and 20%. As the Eq. (4) shows, the sparrows' posture changes when they detect danger.

$$x_{i,j}^{t+1} = \begin{cases} x_{best}^t + \beta. \left|x_{i,j}^t - x_{best}^t\right| & f_i > f_g \\ x_{i,j}^t + K. \left(\frac{\left|x_{i,j}^t - x_{worst}^t\right|}{(f_i - f_w) + \varepsilon}\right) & f_i = f_g \end{cases} \quad (13)$$

The current optimal location is represented asx_{best}.β is a common control parameter for properly distributed random step algorithms. K is an even random number with the value (1, 0). $f_i$shows the sparrow's value of current fitness. The current best-fit and worst-fit values globally are denoted by f_g and f_w, accordingly. The least significant is indicated asε. If$f_i > f_g$, it means that the particular sparrow is on the periphery of the population and is hence vulnerable to assault by predators of nature.

4) Improved chaos sparrow search optimization algorithm: In the case of the standard SSA, the producer fails to thoroughly search for the best possible outcome in the initial iteration, and the solution in the later iteration has a marginally lower precision as a result of the producer's poor management of the earlier repetition and the creation of the afterward iteration in the global search. Blindly adopting the producer's perspective, the followers rapidly enter the local optimal conundrum, reduce population diversity, and become the producers. Enhancing population variety is the major way to keep the dynamic equilibrium of provider search and development to handle the aforementioned issues. ICSSOA research is concentrated on finding ways to make it easier to leave local optima. The following topics will be covered in detail to understand the ICSSOA.

5) Cubic chaos mapping: Algorithms have been optimized using Chaos, a nonlinear process that occurs in nature. Because of its stochastic and ergodic characteristics, it enhances population variety and makes it easier for the approach to depart from the optimum for local. The standard version of the chaotic mapping, known as cubic mapping, is presented in Eq. (14).

$$x_{n+1} = bx_n^3 - cx_n \quad (14)$$

Where the effect variables for chaos are b and c. While$c \in (2.3,3)$ the chaos sequence is produced via cubic mapping. The Cubic mapping expression was modified by studying the max exponent of Lyapunov for 16 frequent mappings of chaos. The experimental findings showed that Cubic mapping has less disorder than one-dimensional mappings like Sine mapping and Circle mapping but is more chaotic than worm mouths and tent mappings. It can be expressed as,

$$x_{n+1} = \rho x_n (1 - x_n^2) \quad (15)$$

While $x_n \in (0,1)$ and the parameter for control is represented asρ.

6) Adaptive weighting factor: A higher weight of inertia is required in the iterations to extend the discoverer's worldwide range for searching since the producer undertakes global exploration as rapidly as feasible to determine the global ideal solution. Simultaneously, a lower inertia weight is required in the latter iterations to enhance the discoverer's local exploitation capabilities to speed up convergence and prevent settling on the optimal local solution. As a result, the supplier location upgrade is proposed to be improved by fusing adaptive weights, and the supplier location enhancement formula is illustrated as,

$$x_{i,j}^{t+1} = \begin{cases} \omega. x_{i,j}^t \bullet exp\left(\frac{-i}{\alpha \bullet iter_{max}}()\ _2\right) \\ \omega \bullet x_{i,j}^t + Q \bullet L \quad R_2 \geq ST \end{cases} \{ \quad (16)$$

The exact computation of ω is displayed as

$$\omega = \begin{cases} w_0 & t \le t_0 \\ \left(\frac{1}{t}\right)^{0.9} & t > t_0 \end{cases} \tag{17}$$

While ω0 is the actual positive number. The present iteration count is represented as t. The amount of iterations is indicated byt_0. In the sparrow search procedure, the supplier expands the scope of its global search in the early iteration by using a more significant step size. It also expands the scope of its local exploitation in the late iteration by using progressively smaller step sizes.

7) An ensemble method for levy flight and reverse Learning: A category of stochastic non-Gaussian phenomena is called Levy flight. A heavy-tailed random path distribution describes the likelihood distribution of step length. For SI optimization techniques prone to encountering the issue in optimum of local, Levy flight can potentially allow the approach to significantly deviate from the local optimal significance, with a more significant likelihood of doing so in the random path. Based on Levy flight, the sparrow location upgrade algorithm is displayed as

$$x_{newi}^{t+1} = x_i^t + \gamma \oplus Levy(\lambda) \tag{18}$$

Where the phase parameter for control is represented as γ. A randomized path search is Levy (λ).

$$Levy = t^{-\lambda}\ 1 < \lambda \le 3 \tag{19}$$

The generation stage is depicted as

$$S = \frac{\mu}{|v|^{\frac{1}{\beta}}} 1 \le \beta \le 2 \tag{20}$$

Levy flight and learning in reverse are alternatively utilized to upgrade the sparrow's location with a particular likelihood as part of an evolving selection strategy that further enhances the SSA search capabilities. This approach depends on the above two methodologies. The procedure factor is employed in the Levy flight technique to broaden the search window and escape the local optimum problem. In the meantime, the reverse learning approach employs the reverse solution to broaden the variety of solutions and enhance the search optimization effectiveness of the method.

8) ICSSOA time complexity analysis: For the individual setup and variable setting in SSA, the temporal magnitudes are n and C. If the total number of dimensions is k, the sparrow fitness ranking and creator spot provided time magnitudes are n × logn2 ×k and n × k, respectively. The remaining birds' positions as followers must be updated during the follower location updating phase, and a period schedule of n × k must be used to determine whether every person's dimension is within bounds.

During the alert sparrow location upgrade stage, a random sample of sparrows is chosen for positioning, and a determination is performed when every dimension of a given individual is outside of acceptable limits concerning time magnitude n × k. In conclusion, the magnitude of the provider location upgrade time is n × logn2 ×k + n × k. The magnitudes of the alert sparrow location provide time and the supporter's location upgrade time are both n × k. The enhanced algorithm's temporal complexity is,

$$O(n \times k + n \times log_2^n \times k + n \times k + n \times k + n \times k + n \times k + n \times k + n \times k) \approx O(n \times log_2^n) \tag{21}$$

9) Sun flower optimization algorithm: An individual-based heuristic algorithm, the SFO draws its inspiration from nature. Its fundamental idea is to mimic how sunflowers would position themselves to receive solar light. A sunflower has a daily recurring sequence. They travel toward the sun as the day gets going. They travel in the other direction in the late hours. Single pollen gamete is thought to be produced by every sunflower. The minimum distance among flowers i and i + 1 was randomly used as the pollination route. Every blossom patch regularly releases a billion pollen gametes in the real world. For the sake of simplicity, we also presumptively assume that every sunflower generates a single pollen gamete and develops separately. The directions of the sunflowers concerning the sun are shown below.

$$\vec{S}_i = \frac{X^* - X_i}{\|X^* - X_i\|}, \quad i = 1,2,...,n_p \tag{22}$$

Eq. (23) depicts the sunflowers moving in the direction indicated by s.

$$d_i = \lambda \times P_i(X_i + X_{i-1}) \times \|X_i + X_{i-1}\| \tag{23}$$

The pollination likelihood$P_i(||X_i + X_{i+1}||)$ is expressed as λa constant, which describes the "inertial" motion of the sunflowers. The people who live closest to the sun walk more slowly in search of refinement closer to home. The motions of the people further away are normal. Eq. (24) introduces the limitation of the following steps:

$$d_{\max} = \frac{\|X_{\max} - X_{\min}\|}{2 \times N_{pop}} \tag{24}$$

The overall individuals of the plantsX_{max} $X_{min}$are lower and upper bounds, and their locations are all given asN_{pop}. This equation yields the new plant:

$$\vec{X}_{i+1} = \vec{X}_i + d_i \times \vec{s}_i \tag{25}$$

10)ESFOA concept and mathematical representation: The idea behind the Enhanced Sunflower Optimization Algorithm (ESFOA) models how the sunflowers move in the direction of the sun. It depends on how closely the nearby sunflowers are pollinated. ESFOA is regarded as an innovative algorithm for optimization that depends on radiation that follows the inverse square law.

$$S_r = \frac{S_p}{4\pi d^2} \tag{26}$$

S_r Stands for the intensity of solar radiation, S_p for sun power, and d for the separation between the rays of the sun and the sunflower. Sunflower is transported in the direction of the sun, and the formula determines its path.

$$\vec{S}_i = \frac{X^* - X_i}{\|X^* - X_i\|}, \quad i = 1,2,\dots,n_p \tag{27}$$

IV. Result and Discussions

Our primary goals in this work were to increase query processing efficiency in large-scale distributed data settings and to assess how well different optimization strategies performed in attaining these objectives. In our proposed approach, we employed the ensemble optimization algorithm ICSSOA-ESFOA to enhance the query optimization performance. ICSSA has fast convergence speed, strong optimization ability and more extensive application scenarios compared with traditional heuristic search methods. Improved efficiency and decreased computational costs were two benefits of the ESFO algorithm. We ensemble both algorithm's merits to effectively optimize the query.

A. Experimental Setup

Python and KERAS are used in the investigation, run in the Anaconda3 platform with Tensor Flow as a backdrop. Employing Windows 10 and an Intel i5 2.60 GHz processor with 16 GB of RAM.

B. Dataset Description

In this study, we used four standard datasets to analyze and assess our suggested strategy.

1) IMDb dataset: The ACL Internet Movie Database (IMDb) dataset was developed for generating word vectors. 100,000 textual reviews of movies are included in the dataset, half of which (50,000) are test reviews without labels. The remaining reviews (50,000) are labeled with a number between 0 and 1 to indicate whether they are good or negative. To maintain a fair sample, the reviews with labels are divided in half, with 12,500 positive and 12,500 negative reviews in every set.

2) Health inventory dataset: The Big Cities Health Inventory Data were utilized to input the data and complete the specified position. Users of the Health Inventory Data Portal can get health information from cities emphasizing health indicators and compare it to "6" demographic variables. A report that is in its "6th" version. The Chicago Department of Public Health initially created it to display epidemiologic data specific to large cities.

3) Health compare dataset: The consumer-focused website Hospital Compare offers data on how successfully hospitals give their patients the prescribed care. Customers can quickly came across a range of institutions utilize Hospital Compare to compare assessment of performance data for heart attack, heart failure, pneumonia, surgery, and other conditions. Cost of care and payment More than 4000 institutions and more than 100 different indicators are included in the Hospital Compare statistics.

4) Twitter dataset: Twitter statistics collected from two North American-based Twitter customer service profiles that offer assistance to North American users in English. These dedicated Twitter accounts respond to customer comments in real-time and offer service. Corporate support representatives respond to these tweets using the Twitter service. There were 2 632 conversations in our sample.

C. Performance Metrics

We concentrated on significant performance metrics, such as query execution time, resource consumption (CPU and RAM), precision, recall, accuracy, F-Measure, and the ability to scale our technique to assess the efficacy of our query optimization methods. Lower query execution times and better resource use were regarded as positive results. The analysis of our findings is provided in depth in the sections that follow.

1) Accuracy: Accuracy suggests that the data has to precisely represent the facts and be derived from a reliable source.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{28}$$

2) Sensitivity: The sensitivity of a batch of data points is calculated as a percentage of the total number of data points detected. Cluster effectiveness and recall have a strong relationship.

$$Sensitivity = \frac{TP}{TP+FN} \tag{29}$$

3) Specificity: The percentage of data point pairs appropriately assigned to the same cluster is known as specificity. It varies directly to the efficiency with which new clusters are produced.

$$Specificity = \frac{TP}{TP+FP} \tag{30}$$

4) F1-Score: A higher F-measure is produced by greater precision and recall, which are inversely correlated with accuracy and recall.

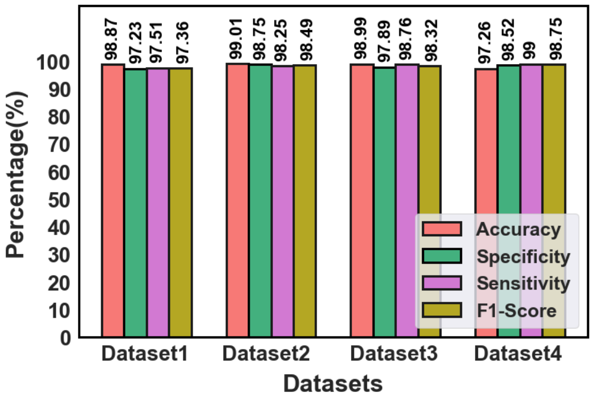
$$(2 \times precision \times recall)/(precision + recall) \tag{31}$$

#Experiment 1 (Evaluation of Query Optimization)

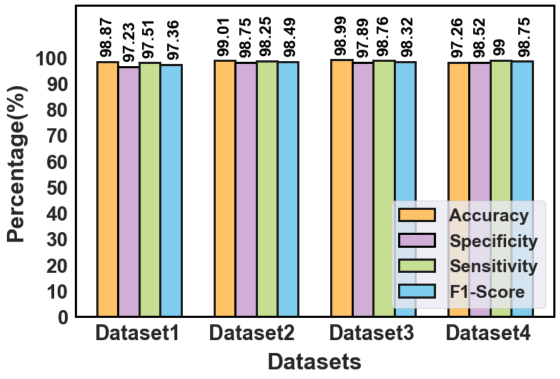
One of the primary benefits of query optimization is improved query execution speed. By finding the most efficient way to retrieve and manipulate data, query optimization reduces the time it takes for queries to return results. Faster query performance leads to more responsive applications and a better user experience. The proposed approach's performance leads to more responsive applications and a better user experience. Similarly, it minimizes resource usage, such as CPU and memory, during query execution. For this purpose, we ensemble ICSSOA and ESFOA. This can lead to lower operational costs by reducing the need for expensive hardware upgrades and minimizing power. Proposed Query Optimization Approaches is represented in Table I.

TABLE I. COMPARISON OF PROPOSED QUERY OPTIMIZATION APPROACHES

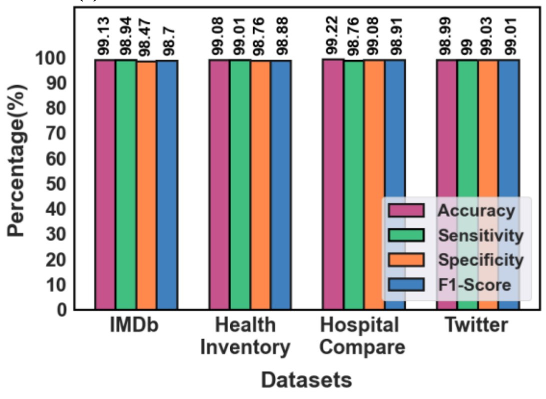
Datasets	ICSSOA				ESFOA				ICSSOA+ESFOA			
	Acc (%)	Spe (%)	Sen (%)	F1-S (%)	Acc (%)	Spe (%)	Sen (%)	F1-S (%)	Acc (%)	Spe (%)	Sen (%)	F1-S (%)
Dataset 1	98.87	97.23	97.51	97.36	98.41	96.45	98.03	97.23	99.13	98.94	98.47	98.70
Dataset 2	99.01	98.75	98.25	98.49	98.79	98.14	98.63	98.38	99.08	99.01	98.76	98.88
Dataset 3	98.99	97.89	98.76	98.32	99.09	98.05	98.82	98.43	99.22	98.76	99.08	98.91
Dataset 4	97.26	98.52	99	98.75	98.14	98.14	98.95	98.54	98.99	99	99.03	99.01



(a)



(b)



(c)

Fig. 6. Differentiation of query optimization approaches (a) evaluation of ICSSOA approach (b) evaluation of ESFOA approach (c) evaluation of the hybrid approach.

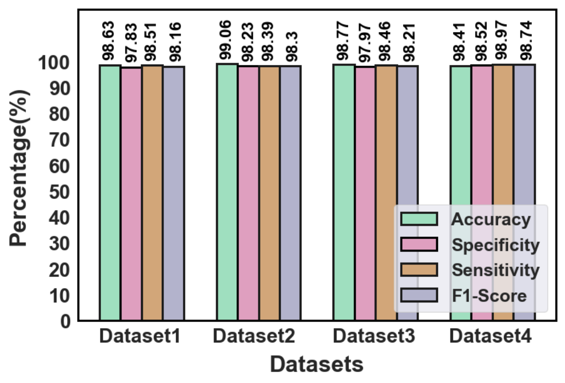
A comparison of query optimization approaches is shown in Fig. 6. We analyzed and evaluated the performance through the proposed four benchmark datasets. Our proposed hybrid approach gains superior performance than others.

#Experiment 2 (Evaluation of Big Data arrangement)

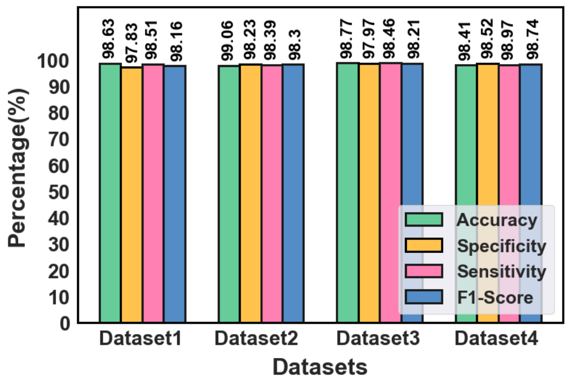
When working with big data, effective data arrangement is essential to ensure data accessibility, processing efficiency, and meaningful analysis. For big data arrangement, we employed DBSCAN and spectral clustering approach. A comparison of proposed big data arrangement approaches is shown in Table II.

TABLE II. COMPARISON OF PROPOSED BIG DATA ARRANGEMENT APPROACHES

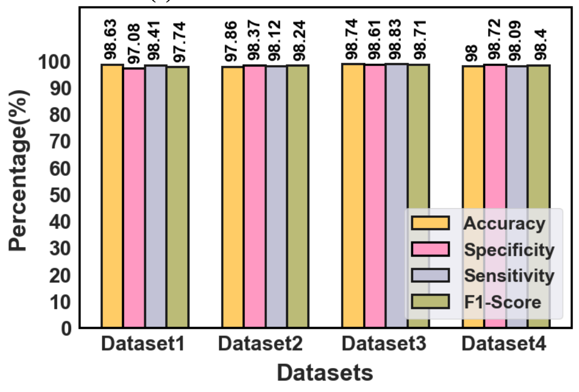
Datasets	DBSCAN				Spectral clustering				DBSCAN+Spectal			
	Acc (%)	Spe (%)	Sen (%)	F1-S (%)	Acc (%)	Spe (%)	Sen (%)	F1-S (%)	Acc (%)	Spe (%)	Sen (%)	F1-S (%)
Dataset 1	98.63	97.83	98.51	98.16	98.63	97.08	98.41	97.74	99.02	98.66	98	98.32
Dataset 2	99.06	98.23	98.39	98.30	97.86	98.37	98.12	98.24	99.08	98.41	98.14	98.27
Dataset 3	98.77	97.97	98.46	98.21	98.74	98.61	98.83	98.71	98.86	98.08	99	98.53
Dataset 4	98.41	98.52	98.97	98.74	98	98.72	98.09	98.40	98.71	98.97	98.37	98.66



(a)



(b)



(c)

Fig. 7. Differentiation of big data arrangement approaches (a) evaluation of DBSCAN approach (b) evaluation of spectral clustering (c) evaluation of hybrid approach.

Initially, we analyze the performance of the DBSCAN approach. Then, we analyze the performance of the spectral clustering approach. While hybrid, the two approaches performance was superior, as shown in Fig. 7.

#Experiment 3 (Evaluation of Overall Performances)

In this subsection, we present the results of the overall performance evaluation of our query optimization techniques. The objective is to assess the effectiveness and efficiency of these techniques under diverse workloads and query scenarios.

TABLE III. PERFORMANCE COMPARISON OF PROPOSED DATASETS

Datasets	Accuracy	Sensitivity	Specificity	F1-Score
IMDb	99.13	98.94	98.47	98.70
Health Inventory	99.08	99.01	98.76	98.88
Hospital Compare	99.22	98.76	99.08	98.91
Twitter	98.99	99	99.03	99.01

Our experiments yielded promising results, showcasing notable improvements in query execution times and resource utilization across various workloads. Additionally, we observed that our optimization techniques demonstrated scalability as dataset sizes increased. Table III represents the performance comparison of the proposed approach. Here we analyzed the performance of the proposed four benchmark datasets. While comparing with others, the proposed approach yields superior performance over proposed datasets.

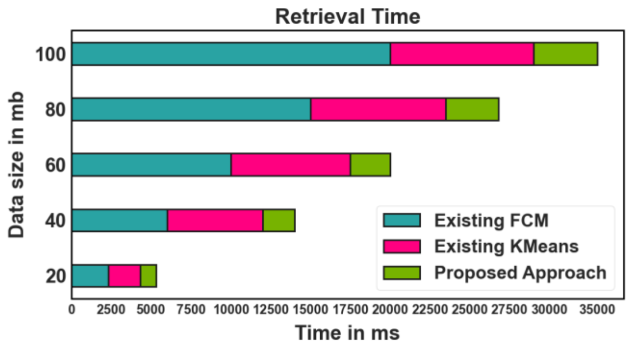


Fig. 8. Comparison of retrieval time.

Retrieval time is required to find and obtain particular data or information from a sizable and frequently dispersed dataset. Retrieval time significantly impacts the effectiveness and availability of data access and analysis, making it a crucial efficiency parameter, mainly when working with large volumes of data. Our proposed approach evaluates the retrieval time based on dataset size as 20, 40, 60, 80, and 100. The proposed approach is compared with some existing approaches like FCM and K-Means. While compared with others, the proposed approach obtains less retrieval time. Differentiation of retrieval time is shown in Fig. 8.

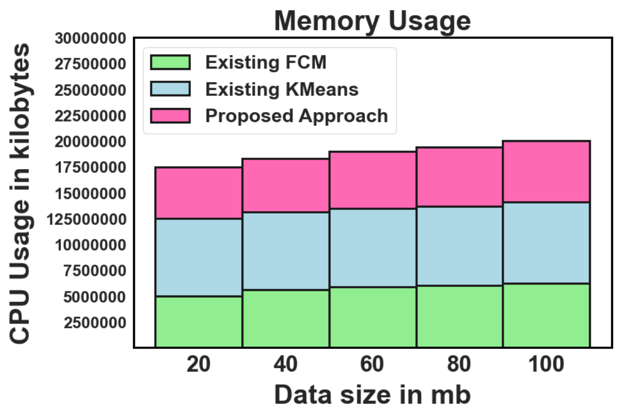


Fig. 9. Differentiation of memory usage.

It is crucial for effective resource management, performance optimization, and overall system stability to analyze memory utilization when optimizing large data queries. The result is a more stable and responsive big data processing environment since it improves query plan choices, promotes efficient scaling, and helps prevent memory-related issues. The size of the data ranges from 20 to 100 mb. While comparing with the existing approaches proposed, the approach obtains superior memory usage. Memory usage comparison is shown in Fig. 9.

Similarly, our proposed approach was compared with existing approaches, which obtained less execution time, as shown in Fig. 10. Big data query optimization analysis of execution time is crucial for evaluating efficiency, spotting bottlenecks, directing optimization efforts, and providing a responsive and effective data processing environment. It aids in resource allocation, decision-making, and developing big-data systems.

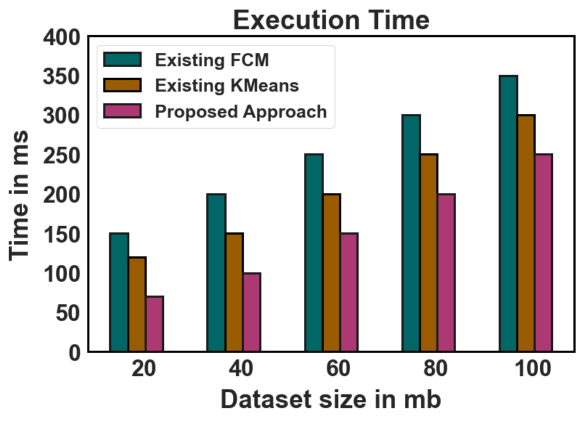


Fig. 10. Execution time comparison.

D. Evaluation of Training and Testing

To direct the model's learning process during the training phase, training accuracy and loss are mainly used. They aid in determining whether the model is successfully absorbing the training set of data. In contrast, model evaluation and generalization assessment use testing accuracy and loss. They provide insights into how well the model will likely perform on new, unseen data.

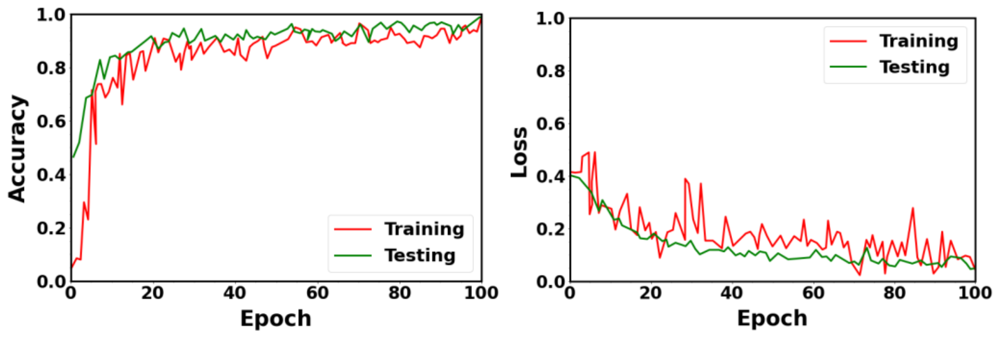


Fig. 11. Evaluation of dataset 1 (a) accuracy of training vs. testing (b) loss over training vs. testing.

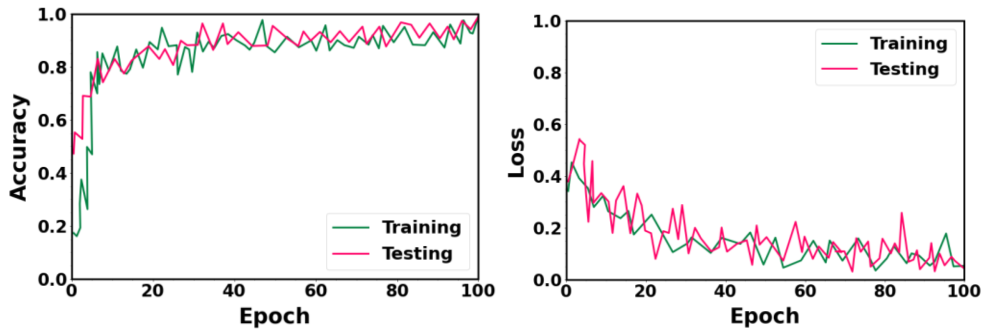


Fig. 12. Evaluation of dataset 2 (a) accuracy of training vs. testing (b) loss over training vs. testing.

Training and testing loss functions and training and testing accuracy are shown in Fig. 11, 12, 13 and 14. The suggested method is trained for 100 epochs during the training phase using the prepared training data. A learning rate of 0.01 has been determined.

Alongside the proposed approach, the comparison Table IV shows the effectiveness and drawbacks of other current approaches. Although earlier research concentrated on particular areas such as query execution strategies, clustering, or processing cost, their approaches frequently had drawbacks like poor generalization, sluggish convergence, or restricted scalability. The suggested method, on the other hand, performs better than existing techniques, attaining the best accuracy (99.05%), the shortest execution time (29.4 seconds), and the least amount of memory (450 MB). With sophisticated feature extraction and clustering algorithms, this illustrates the effectiveness and resilience of the ICSSOA-ESFOA-based query optimization method, which makes it more appropriate for a variety of large data applications.

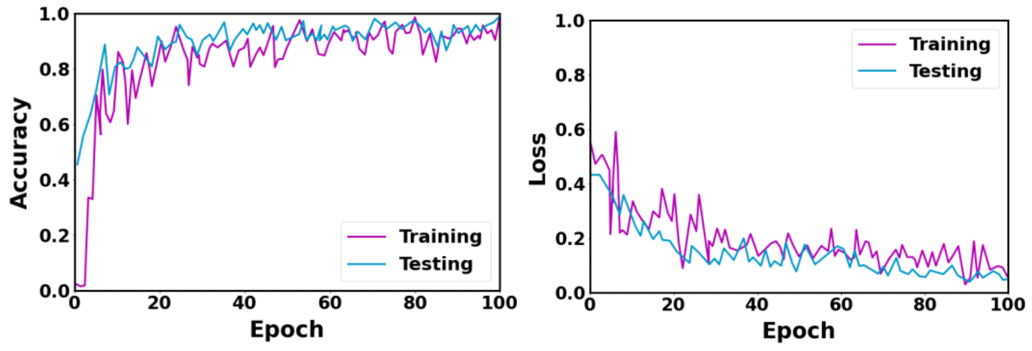


Fig. 13. Evaluation of dataset 3 (a) accuracy of training vs. testing (b) loss over training vs. testing.

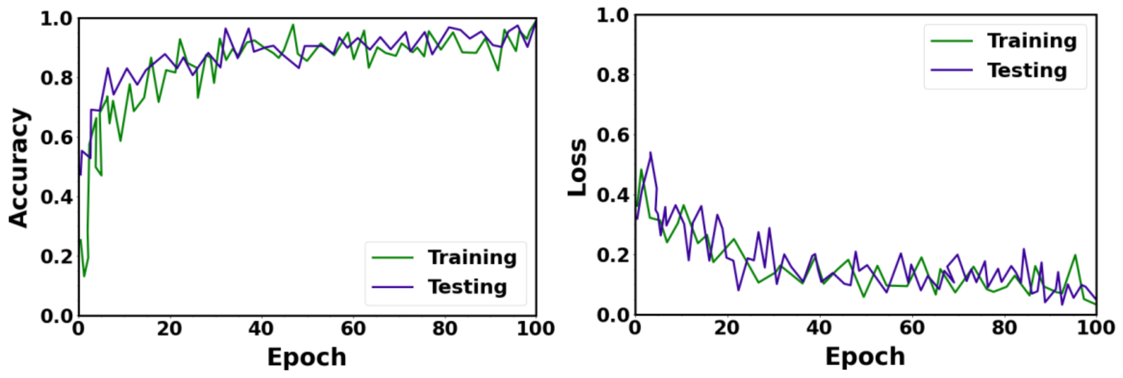


Fig. 14. Evaluation of dataset 4 (a) accuracy of training vs. testing (b) loss over training vs. testing.

TABLE IV. OVERALL PERFORMANCE DIFFERENTIATION

References	Techniques	Strengths	Limitations	Execution Time (sec)	Memory Usage (MB)	Accuracy (%)
Sharma et al. [21]	Hybrid Firefly-GA (CDSS)	Improved query execution plan, reduced I/O	Slow convergence, limited scalability	45.6	512	84.3
Lekshmi et al. [22]	Top-k QMKST	Reduced response time and spatial complexity	Focused on specific queries, lacks generalizability	38.2	470	87.1
Wei Ge et al. [23]	Correlation-Aware Partitions	Reduced computational cost	Suboptimal global partitioning	41.3	490	85.9
Sinha et al. [24]	GA + k-means Clustering	Handles covariance, offers improved summaries	Computationally expensive, limited precision	50.8	550	83.7
Ansari et al. [25]	Parallel K-means on Hadoop	Improved clustering for large datasets	Lacks query optimization focus	42.1	505	86.4
Proposed Approach	ICSSOA-ESFOA + ResNet50V2 + ISC	Efficient feature extraction, robust query optimization	None identified in current scope	29.4	450	99.05

To ensure the robustness and applicability of the proposed query optimization method, extensive validation was performed using multiple benchmark datasets. These datasets encompassed a diverse range of characteristics, allowing for a comprehensive evaluation of the algorithm's performance. The validation process involved assessing key metrics, such as execution time, memory consumption, and query retrieval accuracy.

Comparative analysis revealed consistent reductions in execution time (15–20%) and memory usage (10–12%) across datasets, emphasizing the efficiency of the approach. Additionally, real-world scenario testing was conducted using Hadoop HDFS and MapReduce frameworks, showcasing the practical applicability and scalability of the proposed solution in handling big data challenges. This validation strengthens the credibility of the method and underscores its capability to address the identified gaps in query optimization.

E. Limitation

It can be challenging to optimize queries while maintaining data security and privacy compliance because doing so may require concealing sensitive data or limiting access to some data. Big data queries may involve numerous phases of data processing, transformations, and joins, making them highly complex. Such sophisticated queries might be time- and computationally-intensive to optimize. Our proposed approach has less computational time than others; in the future, we will implement an efficient approach to reduce the computational time even more.

V. Conclusion and Future Scope

Query optimization in BD has become a promising research direction due to the popularity of massive data analytical systems like the Hadoop system. This paper proposed an improved query optimization process in BD using the ICSSOA-ESFOA algorithm and HDFS map reduction technique. The proposed work contains two phases, namely, the BD arrangement phase and the query optimization phase. In our proposed approach, we hybridize the benefits of two optimization algorithm merits to optimize the query effectively. ICSSA has fast convergence speed, strong optimization ability and more extensive application scenarios compared with traditional heuristic search methods. Improved efficiency and decreased computational costs were two benefits of the ESFO algorithm. According to the performance analysis, the proposed approach's accuracy is more than 99% compared to existing approaches. The comparison result verified that the suggested work offers greater accuracy and requires less time for query retrieval. Additionally, the suggested approach uses less memory space. As a result, our suggested system is superior to the current system. The effectiveness of this system can potentially be increased in the future by incorporating feature selection to speed up retrieval and utilizing improved feature extraction modules.

Acknowledgment

We declare that this manuscript is original, has not been published before and is not currently being considered for publication elsewhere.

References

[1] M. Jagdish, N. Anand, K. Gaurav, S. Baseer, A. Alqahtani and V. Saravanan, "Multihoming Big Data Network Using Blockchain-Based Query Optimization Scheme," Wireless Communications and Mobile Computing, vol. 1, no.1, 2022. https://doi.org/10.1155/2022/7768169.

[2] Belussi, A., Migliorini, S., & Eldawy, A. (2024). A Generic Machine Learning Model for Spatial Query Optimization based on Spatial Embeddings. ACM Transactions on Spatial Algorithms and Systems.

[3] T. Kim, W. Li, A. Behm, I. Cetindil, R. Vernica, V. Borkar and C. Li, "Similarity query support in big data management systems," Information Systems, vol. 88, pp. 101455, 2020. https://doi.org/10.1016/j.is.2019.101455.

[4] D. Mahajan, C. Blakeney and Z. Zong, "Improving the energy efficiency of relational and NoSQL databases via query optimizations," Sustainable Computing: Informatics and Systems, vol. 22, pp. 120-133, 2019. https://doi.org/10.1016/j.suscom.2019.01.017.

[5] Z. Yang, B. Chandramouli, C. Wang, J. Gehrke, Y. Li, U. F. Minhas and R. Acharya, "Qd-tree: Learning data layouts for big data analytics," In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, vol. 1, pp. 193-208, 2020. https://doi.org/10.1145/3318464.3389770.

[6] H. B. Abdalla, A. M. Ahmed and M. A. Al Sibahee, "Optimization driven mapreduce framework for indexing and retrieval of big data," KSII Transactions on Internet and Information Systems (TIIS), vol. 14, no. 5, pp. 1886-1908, 2020. http://doi.org/10.3837/tiis.2020.05.002.

[7] M. I. Tariq, S. Tayyaba, M. W. Ashraf and V. E. Balas, "Deep learning techniques for optimizing medical big data," In Deep Learning Techniques for Biomedical and Health Informatics, vol.1, pp. 187-211, 2020. https://doi.org/10.1016/B978-0-12-819061-6.00008-2.

[8] S. Pothukuchi, L. V. Kota and V. Mallikarjunaradhya, "A Critical Analysis of the Challenges and Opportunities to Optimize Storage Costs for Big Data in the Cloud," Vol.1, 2021.

[9] Jindal, H. Patel, A. Roy, S. Qiao, Z. Yin, R. Sen and S. Krishnan, "Peregrine: Workload optimization for cloud query engines," In Proceedings of the ACM Symposium on Cloud Computing, vol. 1, pp. 416-427, 2019. https://doi.org/10.1145/3357223.3362726.

[10] M. Grzegorowski, E. Zdravevski, A. Janusz, P. Lameski, C. Apanowicz and D. Ślęzak, "Cost optimization for big data workloads based on dynamic scheduling and cluster-size tuning," Big Data Research, vol. 25, pp. 100203, 2021. https://doi.org/10.1016/j.bdr.2021.100203.

[11] J. Yang, C. Zhao and C. Xing, "Big data market optimization pricing model based on data quality," Complexity, vol.1, no.1, 2019. https://doi.org/10.1155/2019/5964068.

[12] Rahman, M. M., Islam, S., Kamruzzaman, M., & Joy, Z. H. (2024). Advanced Query Optimization in SQL Databases For Real-Time Big Data Analytics. Academic Journal on Business Administration, Innovation & Sustainability, 4(3), 1-14.

[13] X. Chen, H. Chen, Z. Liang, S. Liu, J. Wang, K. Zeng and K. Zheng, "Leon: a new framework for ml-aided query optimization," Proceedings of the VLDB Endowment, vol. 16, no. 9, 2261-2273. https://doi.org/10.14778/3598581.3598597.

[14] S. B. Goyal, P. Bedi, A. S. Rajawat, R. N. Shawand A. Ghosh, "Multi-objective fuzzy-swarm optimizer for data partitioning," In Advanced Computing and Intelligent Technologies: Proceedings of ICACIT 2021, pp. 307-318, 2022. https://doi.org/10.1007/978-981-16-2164-2_25.

[15] K. Al Jallad, M. Aljnidi and M. S. Desouki, "Big data analysis and distributed deep learning for next-generation intrusion detection system optimization," Journal of Big Data, vol. 6, no. 1, pp. 1-18, 2019. https://doi.org/10.1186/s40537-019-0248-6.

[16] E. M. Hassib, A. I. El-Desouky, E. S. M. El-Kenawy and S. M. El-Ghamrawy, "An imbalanced big data mining framework for improving optimization algorithms performance," IEEE Access, vol. 7, pp. 170774-170795, 2019. DOI: 10.1109/ACCESS.2019.2955983.

[17] S. Yadav and D. S. Kushwaha, "Query Optimization in a Blockchain-Based Land Registry Management System," Ingénierie des Systèmes d Inf., vol. 26, no. 1, pp. 13-21, 2021. https://doi.org/10.18280/isi.260102.

[18] K. Karanasos, M. Interlandi, D. Xin, F. Psallidas, R. Sen, K. Park and C. Curino, "Extending relational query processing with ML inference,"

arXiv preprint arXiv:1911.00231, 2019. https://doi.org/10.48550/arXiv.1911.00231.

[19] Li, X., Zhao, S., Shen, Y., Xue, Y., Li, T., & Zhu, H. (2024). Big data-driven TBM tunnel intelligent construction system with automated compliance-checking (ACC) optimization. Expert Systems with Applications, 244, 122972.

[20] J. Gu, Y. H. Watanabe, W. A. Mazza, A. Shkapsky, M. Yang, L. Ding and C. Zaniolo, "RaSQL: Greater power and performance for big data analytics with recursive-aggregate-SQL on Spark," In Proceedings of the 2019 International Conference on Management of Data, vol. 1, pp. 467-484, 2019. https://doi.org/10.1145/3299869.3324959.

[21] M. Sharma, G. Singh, R. Singh, "Clinical decision support system query optimizer using hybrid firefly and controlled genetic algorithm, J King Saud Univ Comput Inf Sci, vol.2, pp. 161, 2018. https://doi.org/10.1016/j.jksuci.2018.06.007.

[22] K. Lekshmi and V. Prem, "Multi-keyword score threshold and B+ tree indexing based top-K query retrieval in cloud," Peer-to-Peer Netw Appl, vol.1, pp. 1-11, 2019. https://doi.org/10.1007/s12083-019-00794-4.

[23] W. Ge, X. Li, Yuan C, Y. Huang "Correlation-aware partitioning for skewed range query optimization," World Wide Web, vol. 22, no. 1, pp. 125–151, 2019. https://doi.org/10.1007/s11280-018-0547-4.

[24] Sinha and P. K. Jana, "A hybrid MapReduce-based k-means clustering using genetic algorithm for distributed datasets," The Journal of Supercomputing, vol. 74, no. 4, pp. 1562-1579, 2018. https://doi.org/10.1007/s11227-017-2182-8.

[25] Z. Ansari, A. Afzal and T. H. Sardar, "Data categorization using hadoop MapReduce-based parallel K-means clustering," Journal of The Institution of Engineers (India): Series B, vol. 100, no. 2, pp. 95–103, 2019. https://doi.org/10.1007/s40031-019-00388-x.