Iranian Journal of Fuzzy Systems

Volume 22, Number 2, (2025), pp. 1-24

Original paper

University of Sistan and Baluchestan

QPMMCOA and bayesian fuzzy clustering: A novel approaches for optimizing queries in big data

M. Sandhya Rani[1] and N. Raghavendra Sai[2]

[1] *Research Scholar, Department of Computer Science and Engineering, Koneru Lakshmaiah Educational Foundation, Vaddeswaram, Guntur District, 522302, Andhra Pradesh, India.*

[2] *Associate Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Educational Foundation, Vaddeswaram, Guntur District, 522302, Andhra Pradesh, India*

193030044@kluniversity.in, nallagatlaraghavendra@kluniversity.in

Abstract

The explosion of data in the last ten years has led to a substantial focus on big data (BD) in the information area. The philosophical applications of "query optimization (QO)" are crucial in BD environments' data retrieval processes. Several distributed data processing platforms in the cloud were developed to provide BD query optimization services that are both affordable and effective. Nevertheless, due to a lack of consideration for energy-related concerns and query characteristics, most solutions resulted in higher "energy consumption (EC)" and lower accuracy. We introduced an innovative deep-learning approach to arrange big data to overcome the issue. This work presents an effective query optimization that uses the Quantum parallel multi-layer Monte Carlo optimization method (QPMMCOA) optimizer and a load balancer based on Bayesian fuzzy clustering to address the problems associated with the query optimization process. There are two phases to the suggested technique: (1) Big data arrangement and (2) Query Optimization. The first step arranges BD using preprocessing, feature extraction, feature selection, and deep learning-based Big Data arrangement. The improved Deep Residual Shrinkage Network (IDRSN) algorithm is used for the BD arrangement. The essential features are selected using the Chaotic Vortex Search algorithm (CVSA). During the second phase, a Bayesian fuzzy clustering-based load balancer is used with the QPMMCOA optimizer to improve overall query processing performance and ignore energy-efficient query plans. At last, the process of evaluating similarity is carried out. The experimental results demonstrated that the method performed better than other existing algorithms.

Keywords: Query optimization, QPMMCOA, improved deep residual shrinkage network (IDRSN), chaotic vertex search algorithm (CVSA), bayesian fuzzy clustering.

1 Introduction

Today's civilization is experiencing an explosion in data, and the processing and analysis of these massive datasets is known as big data (BD) [15]. Large amounts of data can come from various sources [1, 24]. The annual growth rate of data is estimated to be 1.2 zettabytes. However, the impact of data abundance goes far beyond quantity; the vast number of sources results in a wide variety; seventy percent of data needs to be more organized and will change more quickly than organized data. Processing data on a bigger scale, spanning hundreds of terabytes, is challenging. Accurate, timely, and reliable data acquisition is crucial to the success of Big Data [21, 37, 42]. Managing vast amounts of data to extract useful hidden information has become urgent. Historical data are studied in data warehouses by crafting complex queries; nevertheless.

Maintaining this level of detail in traditional databases is challenging due to the volume of data, the complexity, and the variety of inquiries [12, 46]. Complex and supplementary combination inquiries mainly result in multipart parsing

Corresponding Author: M. Sandhya Rani

Received: December 2023; Revised: February 2025; Accepted: April 2025.

https://doi.org/10.22111/ijfs.2025.47412.8350

trees, which increase the number of searches and total execution time. To reduce both query processing duration and turnaround time, it is necessary to look at the Economic Quality order (EQO), which will result in thousands fewer inquiries [23]-[45]. When considering unstructured data, the intricacy of the data becomes apparent during the analysis process [44]. Large application sectors need accurate and quick responses to inquiries posed on big data. Processing and optimizing big data queries is still in its infancy [33].

A query is an assertion or group of statements to perform the Read, Write, Delete effectively, and Update database operations [26]-[47]. It is essential for the organizing and retrieval of data. It offers guidelines for creating a response to a series of questions with group assignments [11]. To offer more effective and reasonably priced choices for processing BD queries, numerous cloud-based distributed data processing technologies [9]. Various query strategies are used in many relational database management methods to satisfy the queries. An effective query plan is also observed, as it minimizes using specific resources, such as input/output (I/O), by selecting the optimal query access plan [5, 25]. Therefore, this research employed fuzzy-based load-balancing and deep-learning approaches for big data arrangement. Automatically developing hierarchical feature representations from unprocessed data is a strong advantage for deep learning models. This is especially helpful in significant data circumstances, where the volume and complexity of the data can make it challenging to define characteristics. Our proposed approach yielded superior performances and obtained less execution, response, and turnaround times.

1.1 Nove1 contribution

The significant key contributions of this research are as follows,

• Preprocessing techniques like tokenization and stop word removal are used to increase the efficacy and efficiency of queries. While stop-word removal increases performance by removing common words that might not significantly add to the query's relevancy, tokenization breaks down requests into smaller sections to facilitate parsing and analysis.

• The essential features are selected using the Chaotic Vertex Search algorithm (CVSA).

• The selected features are fed into the big data arrangement phases. The novel deep learning-based Improved Deep Residual Shrinkage network (IDRSN) is utilized to arrange the big data.

• Bayesian fuzzy clustering is employed for load balancing.

• The queries are optimized using the Quantum parallel multi-layer Monte Carlo optimization method (QPMM-COA).

The article's next sections are organized as follows. In the second part, the previously issued works are briefly explained. Part 3 provides a detailed description of the suggested approach. Part 4 thoroughly simulates the proposed approach. Part 5 concludes with a statement.

2 Related works

The following is a complete description of the relevant approaches using techniques and algorithms for optimizing queries.

To get around HOME overheads, Sahal et al. [36] proposed the Multi-session Big Data Environment (iHOME), which directly stores only indexes of the linked rows rather than complete intermediate outcomes. The eight JOIN query scenarios the suggested iHOME system handles are categorized into three groups. While contrasting with existing techniques, the presented method performs well.

Sahal et al. [35] presented the Multi-Query Optimization utilizing the Tuple Size and Histogram (MOTH) approach to consider reused-based possibilities' granularity. The suggested MOTH system considers the non-uniform distribution of data and non-equal pair size to use the coarse-grained, wholly and substantially reused-based possibilities between queries and prevent redundant calculations. To accurately evaluate the costs associated with reused outcomes, the MOTH system predicts coarse-grained reused-based possibilities while considering delayed storage. Based on the findings of the experiment. Considering the granularity of repeated results has been observed to reduce the multi-query execution time of the proposed MOTH system using the TPC-H benchmark.

An enhanced query optimization procedure for big data (BD) was suggested by Kumar et al. [18], utilizing the HDFS map-reduce and Ant colony Optimization-Genetic Algorithm (ACO-GA) method. The Secure Hash Algorithm 512 (SHA-512) technique is used to determine the input data's hash value (HV), and the HDFS map-reduce function is used to eliminate duplicate data. After that, characteristics are taken out. The normalized K-Means (NKM) method groups relevant information based on entropy computation. The same characteristics are extracted when the BD questions are gathered in the second phase. Next, the ACO-GA technique is used to find the optimal query. At last, the process of evaluating similarity is carried out. The results of the experiments show that the method performed better than other existing algorithms.

Michiardi et al. [27] presented a method for improving the efficiency of scalable data-intensive computing models. By carefully choosing and using typical phrases to meet memory restrictions, this method changed the numerous searches into the most effective ones. This method applied a cost-centric optimization framework related to the multi-choice knap-sack issue to determine practical and effective execution plans. The query's selectivity may specify the range of its benefits. Ge et al. [10] presented a method known as a correlation-aware partition. It introduced the concept of partitioning optimization, which is akin to a geometric curve-fitting problem, to continuous correlating data. Data must be partitioned according to range query boundaries for the best results. When compared to the global one, the local one performed better. Rather than attempting to increase efficiency, bottom-up merging partitioning was applied to improve partitioning further. The current methods use a variety of algorithms designed to optimize queries; nevertheless, the methods cannot address the problems associated with optimizing queries.

Jagdish et al. [15] developed an intimate framework for multi-blockchain algorithm optimization employing big data inquiry to improve the performance of association query processing among numerous multihoming blockchains by using the big data system. This approach adds semantic evidence to the traditional multi-blockchain prototype. It creates a semantic model of multi-blockchain that serves as a foundation for linking inquiries among the many blockchain multihoming systems in big data. Based on this concept, distributed databases contain index arrangements, and a linking index arrangement is presented among the multiple blockchains, with different qualities associated with these blockchains used to improve the efficacy of linking calculations. Bashabsheh et al. [4] present an integrated Harris Hawks Optimizer (HHO) that combines K-mean clustering and the MapReduce architecture to solve diverse data clustering problems. The suggested approach employs the K-means algorithm to address various issues regarding clustering. More particularly, K-means are used as initial answers to the classic Harris Hawks Optimizer HHO. In general, HHO seeks to improve the potential solutions in order to discover the best one. MapReduce is a distributed computing framework that generates datasets employing a parallel program on a cluster. It was used explicitly in the created HHO for parallelization since it provides fault tolerance, load balancing, and data localization.

A query optimization method called the Enhanced Salp Swarm Algorithm (ESSA) was suggested by Sundarakumar et al. [7] in combination with the Modified K-Means Algorithm (MKM) for cluster development. Two steps make up the process: gathering and organizing the data and then extracting the data from the repository. In their investigation of the efficacy of sophisticated query optimization strategies in SQL databases, Rahman et al. [32] concentrate on query rewriting, multi-level indexing, and dynamic query execution plans. Multi-level indexing reduces data retrieval times by about 40%, query rewriting reduces execution times by 35%, and dynamic query execution plans increase resource utilization efficiency by 25%, according to the findings, which show notable gains in query performance. The use of AI to optimize query processing for localized big data in real-time was examined in Sun and Sun [40]. The goal of the study was to manage massive data queries in dynamic contexts by utilizing AI models to increase processing accuracy and efficiency.

2.1 Challenges

• A SmallClient indexing system was developed to handle the enormous datasets and employ pertinent information. Nevertheless, the approach lowers enormous data efficiency and requires constant improvement.

• One type of multiple dimensions database with distributed nodes in its dimensions needs to be fixed. When data from these databases proves essential, extracting it might be challenging. Determining the top-k dominant values is, therefore, a significant task.

• The enormous volume of processed information and the extremely high number of potential attributes make indexing a significant barrier in the setting of data warehouses. The main issue at hand is indexing the ambiguous information and its sources.

• One major obstacle when working with large datasets is the curse of dimensionality. Numerous methods were developed to improve query efficiency in large dimensionality space using hierarchical indexing using the R-tree. The typical datasets get significantly worse as the size of the data increases, regardless of the difficulty of the course of dimensions.

• Applications involving analytical data become highly complex due to the vast amount of data. In businesses, numerous issues arise from gaining expertise and selecting options based on increasing data. Significant obstacles in the design and execution of data include read/write efficiency, access to data, and on-demand file generation. Big data also presents issues with data security, gathering, analysis, storage, and administration.

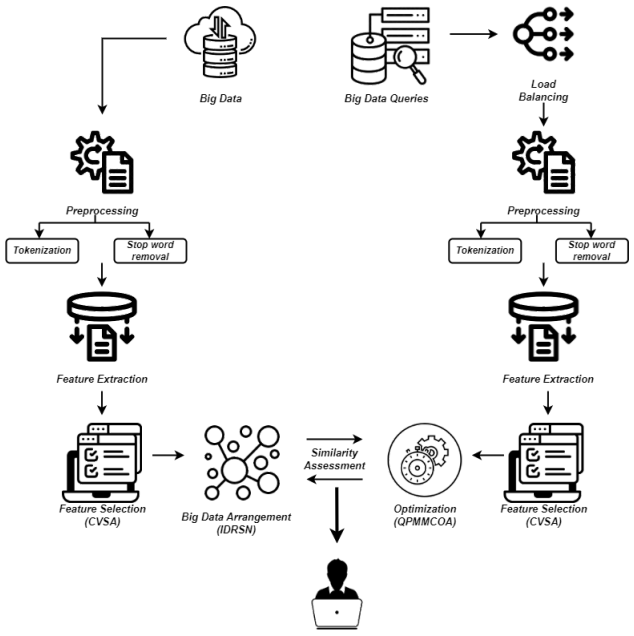


Figure 1: Proposed overall framework

3 Proposed methodology

The proposed method is divided into two main components: Big Data Arrangement and Query Optimization. For Big Data Arrangement, the technique utilizes the Improved Deep Residual Shrinkage Network (IDRSN), which enhances big data processing through an advanced network architecture and optimized training techniques. Essential features are selected using the Chaotic Vertex Search Algorithm (CVSA), which identifies the most relevant data points. In the Query Optimization phase, the Bayesian fuzzy clustering approach is employed to efficiently balance the computational load, while the QPMMCOA fine-tunes the queries for improved accuracy and speed. This method integrates these elements to optimize both data arrangement and query processing, as shown in Figure 1.

3.1 Phase 1: Big data arrangement

According to its definition, big data requires new, higher-performance processing due to its vast volume and velocity. It needs an extensive computer infrastructure to ensure successful data processing and analysis while tackling BD, which is a challenging and laborious project. For better data analysis and faster processing, the BD is organized, in this case, in a well-structured format. The steps for data deduplication, extraction of features, and big data arrangement are all included in this phase of the process.

3.1.1 Data preprocessing

It is the first stage since it improves the accuracy of pattern recognition and speeds up the other procedures needed to extract the recurrent pattern. Two preprocessing approaches used in big data query optimization are tokenization and stop word removal, which aim to increase text-based queries' efficacy and efficiency. These stages are explained in the following sections.

3.1.2 Tokenization

This is essentially the process of breaking the text into a collection of significant small pieces known as tokens. Word boundaries are the focal point of this process. Tokenization's primary benefits include reducing the risk of disclosing private information and eliminating unnecessary tokens.

3.1.3 Stop word removal

It typically alludes to the most frequently used words in a language, such as "the," "is," "but," "at," etc. They don't include any critical information. As a result, the stop words were removed from this dataset to improve its substance.

The proposed approach removes 150 stop words during the preprocessing stage. This helps to exclude frequent, less valuable words from the data, increasing the effectiveness of feature extraction and selection by focusing only on the most relevant phrases.

3.2 Feature extraction

Following preprocessing, the essential characteristics of the real dataset are retrieved, including lift, support, confidence, and frequent item datasets. In the end, the enthalpy amount of these characteristics is calculated and used to generate QO.

3.2.1 Closed frequent itemset

It is a repetitive item that covers the same support as the itemset but lacks an additional set. Closed item sets help prevent data regarding support values from being dropped while removing redundant data sets. From the frequent itemset, the closed frequent set of items is derived. The frequently used item can be represented as

$$F^{''}(D_i) = \{F^{''}(D_1), F^{''}(D_2), F^{''}(D_3), \cdots, F^{''}(D_n)\}. \tag{1}$$

According to Eq. (1), the frequency of the itemset is implied by $F^{''}(D_i)$. Currently, the closed frequently used item has been selected, and it can be expressed below,

$$F_c^{''}(D_i) = \{F_c^{''}(D_1), F_c^{''}(D_2), F_c^{''}(D_3), \cdots, F_c^{''}(D_n)\}. \tag{2}$$

The closed frequently used item is implied by $F_c^{''}(D_i)$ Equation (2).

3.2.2 Support

It is among the metrics used to gauge interestingness. This explains the value and assurance of rules. As a proportion (%) of each transaction, it measures how frequently the combination of items P and Q appears together. The following specifies the support of an association rule $P \to Q$:

$$\text{Support}(p \to Q) = \rho(P \cup Q). \tag{3}$$

While the likelihood of the union of the sets of items P and Q is implied by $\rho(P \cup Q)$.

3.2.3 Confidence

It calculates each item in Q shows up in expenses, including items from P. This is estimated using a constrained capability expressed in support for item sets. An association rule $P \Rightarrow Q$'s confidences are given by,

$$\text{Confident}(P \Rightarrow Q) = \frac{\text{Support}(P \cup Q)}{\text{Support}(P)}. \tag{4}$$

3.2.4 Lift

The proportion of absolute confidence to anticipated confidence is known as lift. A rule of associations $P \Rightarrow Q$'s lift is defined as

$$\text{Lift}(P \Rightarrow Q) = \frac{\text{Confidence}(P \Rightarrow Q)}{\text{Support}(Q)}. \tag{5}$$

The sets of items P and Q's implication are implied by $\text{Lift}(P \Rightarrow Q)$ in Equation (5).

3.2.5 Entropy

Entropy measures the data trait's mean. It has nothing to do with how distributions of likelihood vary. Entropy is the arbitrary variable produced by the data quantity of each characteristic. This variable may be the mean or anticipated value.

$$E_y(S,C,L) = -\sum_{i=1}^{n} \rho(S,C,L)_i \log \rho(S,C,L)_i. \tag{6}$$

Equation. (6) represents the likelihood of lift, confidence, and support$E_y(S,C,L)$ variations. These entropy values represent the distribution stage for a BD arrangement.

3.3 Feature selection

In big data query optimization, feature selection refers to selecting a subset of pertinent features from a vast range of potential features to increase the effectiveness and efficiency of queries. Queries can be improved to execute faster and use less computing power by concentrating on the most essential features. We used the Chaotic Vertex Search algorithm (CVSA) for feature selection. CVSA can effectively handle high-dimensional search spaces. When used in big data contexts, where datasets frequently contain many features, CVSA can efficiently explore the solution space and find pertinent feature subsets for optimization. The CVS method takes an innovative approach to feature selection in query optimization, leveraging chaotic dynamics to improve search efficiency and efficacy. CVS provides various search processes by using chaotic maps, allowing for a more complete exploration of the feature space and avoiding local optima. This adaptive search approach excels in handling high-dimensional and complex data, resulting in the identification of the most relevant features for improvement. CVS also minimizes computational costs by efficiently limiting the feature set, and its stable performance in a variety of contexts makes it appropriate for a wide range of query optimization applications.

CVSA, for feature selection, employs chaotic dynamics to optimize the search for the most relevant features. CVSA, unlike standard approaches, generates different and non-repetitive search trajectories using chaotic maps. This method aids in thoroughly exploring the space of features and avoiding local optima, resulting in a better selection of characteristics that are most relevant for the optimization process. CVSA successfully balances exploration and exploitation by utilizing chaotic behaviour, resulting in more accurate and efficient discovery of critical traits.

3.3.1 Chaotic vortex search algorithm

The vertical flow of the agitated fluids causes the VSA, an innovative metaheuristic optimizing algorithm, to shift into the stimulation mode. Like other single-solution algorithms, it uses shortened generation stages in its operations. Using numbers, the VSA population's generations are changed to any era, entirely shaping contemporary solutions. Additionally, a crucial component in presenting one solution is the efficiency of each upgrade and seek of the iteration step at the seek space. This equilibrium is achieved within the suggested VSA with the help of a vortex-like searching pattern. Most of the layered circles mimic the vortex specimen techniques.

3.3.2 Generating the initial solution

First, "centre" is represented as (μ_0) and "radius" is denoted as (r_0) is initialized. Equation (7) can be used to determine the initial centre (μ_0) in this stage.

$$\mu_0 = \frac{upper\ limit + lower\ limit}{2}. \tag{7}$$

The problem's binding restrictions, upper and lower limits, can be expressed as vectors in $d \times 1$-dimensional space. Furthermore, σ_0 is the starting radius r_0 that is produced by equation (8)

$$\sigma_0 = \frac{\max(upper\ limit) - \min(lower\ limit)}{2}. \tag{8}$$

3.3.3 Generating the candidate solutions

When generating individuals Ct(s) in any repetition, t being the t^{th} iteration, developing potential solutions is employed. The VSA is created randomly around the initial centre.$\mu_0$using a Gaussian distribution. Multivariate distribution with

Gaussian formula (9) is displayed.

$$\rho\left(x \mid \mu, \sum\right) = \frac{1}{\sqrt{(2\pi)^d \mid \sum \mid}} \exp\left\{-\frac{1}{2}(x-\mu)^{T^{-1}} \sum (x-\mu)\right\}. \tag{9}$$

In equation (9), the value of the dimension is denoted by d, the random variable's $d \times 1$ vector is represented by x, and the set of samples mean's $d \times 1$ vector is denoted by μ. $\sum$ represents the matrix of covariance. According to formula (4), the range will take on a sphere-like form when the diagonal components of the $\sum$ numbers are identical and the off-diagonal elements are equal to zero. Therefore, using identical variances with zero covariance, the value of $\sum$ is calculated.

$$\sum = \sigma^2.[I]_{d\times d}. \tag{10}$$

3.4 Replacement of the current solution

To facilitate the selection process, the existing solution is changed. Confirming that the suggested solutions are within the identified spaces (Eq. (11)) before the selection procedure is necessary.

$$S_k^i = \begin{cases} rand \cdot (upper\ limit^i - lower\ limit^i) + lower\ limit^i,\ s_k^i < lower\ limit^i, \\ rand \cdot (upper\ limit^i - lower\ limit^i) + lower\ limit^i,\ s_k^i > upper\ limit^i. \end{cases} \tag{11}$$

In the case of a randomly distributed randomized number, $k = 1, 2, \cdots, n$ and $i = 1, 2, \cdots, d$ and rand denoted. The next set of solutions is chosen to utilize VSA with s as the updated centre and reduce the vortex size employing Eq (11). As a result, it is possible to create a novel set of possibilities C1(s). A strategy can be decided to be the updated best solution and committed to memory if it is superior to the best solution.

3.4.1 The radius decrement process

An inverted imperfect gamma function reduces the diameter number on every subsequent pass in the VSA. In probability, the partial gamma function given in Equation (12) is frequently encountered, particularly in applications about the chi-square probability.

$$\gamma(x, a) = \int_0^{\cdot\cdot} e^{-t} t^{a-1} dta > 0. \tag{12}$$

Meanwhile, the value of the random variable $x \geq 0$ and the shape of parameter $a > 0$ are present. Equation (13) also typically introduces its supplementary, $\Gamma(x, a)$, comparable to the imperfect gamma function.

$$\Gamma(x, a) = \int_0^{\infty} e^{-t} t^{a-1} dta > 0. \tag{13}$$

3.4.2 CVSA

The hybrid version of VSA and CMs will be described in this section. Chaos Maps (CMs) are algorithms that generate chaotic sequences to improve search performance. These maps introduce randomization, allowing the algorithm to explore the feature space more fully while avoiding local optima, resulting in better feature selection. The centre and radius are two crucial components of the VSA's straightforward design. Initially, the centre is the present point from which the issue arises. A space of search where repetition skips can be used to evaluate the VSA. To replace a new population position, VSA employed this function to find the "centre" in the search for the best solution that has been conducted thus far. Second, "radius" is a technique used to reduce the complexity of the problems; it turns a vast radius problem into a tiny radius problem.

Furthermore, a VSA called a Gaussian distribution is employed to stabilize the exploration and extraction process at each skip in the cycle. To produce potential answers to the current great response, the VSA employed a single centre, which is referred to as the single strategy, which is better than any other. Nonetheless, the drawbacks of the VSA are not evident when considering the local component, as it experiences problems with multiple neighbourhood minimal values. Using a Gaussian distribution, the radius used to update the pleasant solution also reduced the new release skip, making it easier to trap the VSA in nearby optima.

One describes chaos as a phenomenon that occurs. Any exchange of its initial scenario may also aim to modify future behaviour non-linearly. Among the optimization concepts for algorithms for searching is chaos optimization. Its

main idea is to drastically alter the parameters and variables from the chaotic to the solution-focused region. It depends on examining chaotic motion features outside the global optimum, such as ergodicity, regularity, and stochasticity. The two main benefits of chaos are its ability to prevent local minima and its quick convergence. CMs possess a definite form where no random elements are incorporated.

3.4.3 Fitness function

Each point's spot is assessed at each iteration using a unique fitness function fit. Specifically, testing and training datasets are randomly separated into remarkable components using m-fold algorithms. In the function of fitness calculation that follows, a factor for weight is set as in Equation (14) to combine the two norms into one.

$$Fit = maximize\left(a + \beta \times \left(1 - \frac{L_n}{L_s}\right)\right). \tag{14}$$

L_n is the overall quantity of characteristics, β is the weighting factor with an integer in $[0, 1]$, and L_s is the total length of the feature that was chosen as a subset. The total amount of selected characteristics and their significance are controlled by β. For any algorithm, increasing accuracy is the main objective. Hence, values close to 1 are typically used for the weight factor, which β was fixed to 0.8 in this paper. Maximizing classification accuracy while minimizing the amount of chosen characteristics is the optimal strategy.

The initial setup of the metaheuristic algorithm typically involves defining parameters such as population size (usually between 20 and 100), number of iterations (commonly 50 to 500), crossover rate (usually 0.6 to 0.9), mutation rate (generally 0.01 to 0.1), and search space boundaries (which vary based on the problem).

3.5 Big data arrangement employing IDRSN

Deep learning-based methods for arranging massive data sets fulfil several objectives by taking advantage of neural networks' inherent capacity to recognize and convey intricate patterns across vast and heterogeneous datasets. We employed the Improved Deep Residual Shrinkage Network (IDRSN) approach.

3.6 Deep residual shrinkage network

Using a cross-layer link, the DRSN has a higher capacity and a more significant learning impact than the typical deep network. Batch Normalization (BN), global averaging (GAP), activation function, Convolutional layer, and cross-entropy error function are the components of the DRSN. Concurrently, this model has an attribute called the attention mechanism. Depending on the needs of the target, it can assign varying attention weights to various aspects. The mechanism considers other aspects as noise and either eliminates them or lowers their weights compared to the target characteristic. The following is the expression for the Convolutional layer's principle:

$$Y_j = \sum_{i \in M_j} X_i \times K_{ij} + B_j. \tag{15}$$

The input map of the feature is denoted by x, and the resultant map of the feature is indicated by y. The kernel of Convolutional is denoted by k, the bias is represented by b, the channel indices are i and j, and the channel collection is used to calculate Y_{jis} represented by M_j.

BN is a training method for deep learning networks that normalizes input features. BN primarily consists of two phases: characteristics are first adjusted to the accepted median distribution, and then characteristics are adjusted to the ideal probability. Consequently, BN can enhance the model's development speed by preventing the gradient vanishing issue, reducing internal correlated shift, and speeding up learning resolution. The BN equation is represented by

$$\mu = \frac{1}{N_{batc?}} \sum_{n=1}^{N_{batc?}} x_n, \tag{16}$$

$$\sigma^2 = \frac{1}{N_{batc?}} \sum_{n=1}^{N_{batc?}} (x_n - \mu)^2, \tag{17}$$

$$\hat{x}_n = \frac{x_n - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{18}$$

$$y_n = \gamma \hat{x}_n + \beta. \tag{19}$$

In this case, the nth observation's input is represented by x_n, while its output is represented by y_n. The quantity of a minibatch is represented by $N_{batch}.\epsilon$, a constant approaching zero. Two parameter values are utilized to shift the range: β and γ. γ is employed to scale the range of values. Training can be used to produce both γ and β.

The activation function is an essential neural network mainly utilized for nonlinear transformation. Due to its ability to efficiently avoid gradient vanishing, the rectifier linear unit (ReLU) has found widespread application in neural networks. The linear unit of a rectifier is represented as

$$y = \begin{cases} x, \ x \geq 0 \\ 0, \ x < 0 \end{cases} \tag{20}$$

While x and y represent the input and outcome features.

GAP can compute the mean value from every channel before the FC results layer. There are two ways that GAP is advantageous. GAP can lower the likelihood of the fitting phenomena that arise during neural network training and the total amount of neural network parameters. Additionally, GAP can minimize the impact of fault pulse position on training learning features and eliminate the shift variation issue. A common choice for a loss function is the cross-entropy error, which can boost a neural network's training effectiveness. Before determining the cross-entropy error, the softmax function must be used. By, we may define the function of softmax as

$$y_j = \frac{e^{x_j}}{\sum_{i=1}^{N_{class}} e^{x_i}}. \tag{21}$$

Each j^{th} neuron in the function y_j represents its respective output and input values. A neuron's input is denoted by xi, while the total number of neurons is indicated. N_{class}. The convolution kernel in the residual component can alter the outcome feature map's size. A value in the range of 0 to 1, α is what the sigmoid function measures. Because it doesn't completely turn the output to zero, the threshold is positive but not excessively high. Since every channel in the map of features correlates to a shrinkage threshold, the threshold that the module with residuals determines is a vector rather than a value.

3.6.1 Improvement of the deep residual shrinkage network

The Deep Residual Shrinkage Network (IDRSN) can be improved by refining its architecture and training approaches. This involves adding more layers and improving residual connections, using regularization methods to prevent overfitting, optimizing training with adaptive learning rates, and incorporating advanced feature extraction algorithms. These changes improve IDRSN's capacity to process and organize complex big data. The formula used in (22), which includes the rotation A of x and its constant deviation, is helpful when applying soft thresholds to multiobjective optimization issues. In some cases, edge solutions are lost, and complicated computations result.

$$\underset{X}{argmin} \parallel AX - B \parallel_2^2 + \lambda \parallel X \parallel_1 . \tag{22}$$

After that, the enhanced Lagrangian function becomes,

$$L_\beta(X, Z, Y) = f(x) + g(z) + Y^T(AX - B - Z) + \frac{\beta}{2} \parallel AX - B - Z \parallel_2^2 . \tag{23}$$

Where denotes the globally persistent variable X, a local Y variable, and the punishment function. After separate training for every node, the system eventually converges on a globally consistent result. The iterative updating process has the following form:

$$X_i^{k+1} = \underset{X_t}{argmin} \left(\parallel A_i X_i - B_i \parallel^2 + Y_i^{T^k}(X_i - Z^k) + \frac{\beta}{2} \parallel X_i - Z^k \parallel_2^2 \right). \tag{24}$$

$$Y_i^{k+1} = \left(\lambda \parallel X \parallel_1 + \sum_{i=1}^{N} \left(-Y_i^{T^k} + \frac{\beta}{2} \parallel X_i - Z^k \parallel_2^2 \right) \right). \tag{25}$$

$$Y_i^{k+1} = Y_i^k + \beta \left(X_i^{k+1} - Z_i^{k+1} \right). \tag{26}$$

The following outcome can be obtained by updating X_i^{k+1} using the near-end gradient descent approach and by taking the product of x:

$$X^{k+1} = (A_i^T A_i + \beta I)^{-1} (A_i^T b_i + \beta z^k - Y_i^k). \tag{27}$$

Z can be updated by repeating the soft threshold:

$$Z^{k+1} = S_{\lambda/\beta N} \left(\frac{1}{\beta N} \sum_{i=1}^{N} (\beta X_i^{k+1} + Y_i^k) \right). \tag{28}$$

Where,

$$soft_k(a) = \begin{cases} a - k, & a > K \\ 0, & \mid a \mid \leq K \\ a + K, & a < -K \end{cases} \tag{29}$$

The specified soft threshold operator is given by formula (29). Obtaining the local minimum involves a continual near-end gradient drop while the gradient's absolute value approaches zero. The Lipschitz constant, which is connected to the optimization of the L1 norm and requires the transpose of the noise matrix scaled by the highest value of the noise signal, may not be known or calculable throughout the computation with a given step size. The computation's efficiency decreases with large input data sets. This study uses a backtracking function with the fast iterative shrinkage-thresholding algorithm (FISTA) to address the issues above. The following are the primary steps:

$$Q_L(X, Z) = f(Z) + g(X) + \frac{\beta}{2} \parallel AX - B - Z \parallel_2^2 + \langle AX - B - Z, \bigtriangledown f(Z) \rangle. \tag{30}$$

In this technique's iterative computation procedure, the initial location for the subsequent approximating function is the linear sum of the preceding two iterations.

3.7 Phase 2: Query optimization

In this case, QO is carried out to minimize the number of system resources needed to respond to a query and give the user the correct result set in a faster length of time. Second, the system can process more queries in the same amount of time since it takes remarkably less time to consider each request for non-optimized queries.

3.7.1 Load balancing

The cloud system is assigned a specific load based on the BD and user inquiries. Differential system failures and energy loss, such as power consumption, system failure, execution time, etc., are caused by under- and overloaded circumstances. The load balancer distributes the data equitably based on the server's activity level at a given moment. Clients must wait longer for the data to be processed without a load balancer, which may irritate them more. When computing resources are dispersed among several nodes in distributed systems, efficient load balancing is essential to maximizing system performance. Unbalanced workloads can reduce the overall effectiveness of big data processing by causing bottlenecks, longer query response times, and underutilization of available resources. We used Bayesian fuzzy clustering (BFC) for load balancing. BFC makes adaptive load distribution based on system dynamics possible. The load-balancing approach can be dynamically modified to accommodate changes in query workloads, availability of resources, and system constraints.

3.7.2 Bayesian fuzzy clustering

Growing data velocity, variety, and volume provide significant obstacles to effective processing and resource management in big data query optimization. For the distributed computing system to function at its best and allocate resources in such dynamic contexts, load balancing is a crucial component that becomes even more important. Big data systems' inherent complexity and unpredictability may be difficult for traditional load-balancing techniques to handle. This introduction examines how BFC might be creatively applied as a viable paradigm for big data query optimization, specifically in solving load balancing issues.

The Bayesian fuzzy clustering method uses the Bayesian likelihood estimation method to perform the clustering. The BFC approach considers that global optimization plays a part in the objective function's load balancing as nonlinear. Training data T is clustered into B clusters using the Bayesian clustering method. Initializing the Dirichlet condition and choosing the appropriate hyperparameters are necessary before clustering can be carried out. The symmetric Dirichlet distribution is represented as is used in the Bayesian clustering procedure.

$$m_{xy}^{+} \sim \ Diric?let(\delta = 1_B). \tag{31}$$

Thus, the Dirichlet parameter is denoted by δ. Given that the variables used to calculate the likelihood distribution are assumed to remain constant under the Bayesian inference approach, the distribution of conditions for the process of clustering is shown by the following expression:

$$\tilde{P}(T_{xy}, m_{xy} \mid O) = P(T_{xy} \mid m_{xy}, O)\tilde{P}(m_{xy} \mid O). \tag{32}$$

$$\tilde{P}(T_{xy}, m_{xy} \mid O) = \prod_{b=1}^{B} F(T_{xy} \mid O_b, m_{xy\ b}^{f}) m_{xy\ b}^{-f.d/2}\ diric?let(m_{xy} \mid \delta). \tag{33}$$

While T_{xy} and m_{xy} denotes the trained data and matching data point, O indicates the clustering procedure's results, F stands for the distribution operation ffor the fuzzier variable, and d for the data's dimensions. The conditional likelihood obtained in Equation (33) is then used to create the new membership function for the Bayesian fuzzy clustering framework. The following is the function of membership for Bayesian clustering:

$$P = \min\left\{1, \frac{p(T_{xy}, m_b^{+} \mid O)}{P(T_{xy}, m_b \mid O)}\right\}. \tag{34}$$

As stated above, there's no need for the Hastings adjustment for the membership procedure. After applying the distribution conditional and constructed membership functions, the prototypes can be clustered using the Gibbs sampling process. The following represents the groupings associated with the prototypes regarding variation in change and the chain's Markov state:

$$o_b^{+} \sim F\left(o_b, \frac{1}{\beta}.v\right). \tag{35}$$

The parameter for adjusting the tightness is indicated by β and the expression v denotes the variance of that parameter. The expression denotes the clustered specimen for the cluster b o_b^{+} in the formula (35). The following is the final formula for grouping the training data using the cluster prototype and membership degree that remain constant:

$$P(T_{xy}, O_b \mid M) = P(T_{xy} \mid M, o_b)P(o_b). \tag{36}$$

In this case, M represents the number of members in the function's appropriate MAP estimations. The proportion based on a Gaussian proposal expresses the correctness of the equation (36) establishing the clustering samples.O_b. Accordingly, the following circumstance qualifies the clustering equation (36) for Bayesian fuzzy clustering as valid:

$$U_o = \min\left\{1, \frac{p(T_{xy}, o_b^{+} \mid M)}{P(T_{xy}, o_b \mid M)}\right\}. \tag{37}$$

The data used for training the matrix's B clusters are produced using the Bayesian fuzzy clustering approach, and their result is shown in the following manner:

$$b = \{b_1, b_2, \cdots, b_j, \cdots, b_B\}. \tag{38}$$

In which the expression denotes the j^{th} cluster centre b_j.

3.7.3 Preprocessing, feature extraction and feature selection

After it has been gathered, preprocessing, extraction of features, and feature selection are applied to the balanced query data. Finally, the query optimizer uses the obtained entropy values to determine the best queries.

3.7.4 Query optimization

An effective strategy to address the optimization problem is to narrow the feasible zone using a nested partition optimization algorithm. Restricting feasible areas has been used in the linear program narrowing approach to actualize the search outcome. Quantum states superposition of quantum states, qubits, and other quantum physics rules and notions have garnered much interest and attention over the past few decades. To propose a QPMMCOA, this work integrates the idea of narrowing the achievable area approach, the Monte Carlo technique, and quantum bits. The suggested approach continuously reduces the feasible zone for the optimization process. In other words, the QPMMCOA continuously reduces the optimal solution's search space size to get closer to the global ideal solution and produce more precise optimization outcomes.

3.7.5 Mathematical implementation

For constant space with n dimensions optimization issues, a point or vector is typically considered the solution. The definition of a continuous improvement issue is

$$\begin{cases} \min(x) = f(x_1, x_2, \cdots, x_n) \\ a_K \leq x_k \leq b_k;\ k = 1, 2, \cdots, n \end{cases} \tag{39}$$

While the point is represented as x, the function of the objective is defined as f, and n denotes the dimension of space. $[a_K, b_k]$ is the area of the variables x_k. Typically, the vector's fitness value is represented by the value of f. The value of x that makes the $f(x)$ value optimal may be the best option.

3.7.6 Qubit of quantum computing

Qubits can be used in the field of quantum computing to represent data. Typically, the qubit's state is displayed as:

$$\mid \psi\rangle = \alpha \mid 0\rangle + \mu \mid 1\rangle. \tag{40}$$

In this case, both $\alpha\mu$ represent the qubits' likelihood amplitudes.

3.7.7 Initialization of population

The qubit likelihood amplitude is immediately utilized for the individual's setup in the QPMMCOA. Every member of the community is chosen at random. The individual's coding system is articulated as

$$q_j = \left[\begin{pmatrix} \cos(\theta_{j1}) & \cos(\theta_{j2}) \\ \sin(\theta_{j1}) & \sin(\theta_{j2}) \end{pmatrix} \cdots \begin{pmatrix} \cos(\theta_{jn}) \\ \sin(\theta_{jn}) \end{pmatrix} \right],\ j = 1, 2, \cdots, m;\ k = 1, 2, \cdots, n. \tag{41}$$

Every randomized entity in Eq. (41) has two places throughout the traversal space. Maintaining a community's size constant allows for double the investigated space. The search area can be doubled to enhance the global optimum and prevent early maturity. For every random person, two positions are computed as

$$\begin{cases} q_{j0} = (\cos(\theta_{j1}), \cos(\theta_{j2}), \cdots, \cos(\theta_{jn})) \\ q_{j1} = (\sin(\theta_{j1}), \sin(\theta_{j2}), \cdots, \sin(\theta_{jn})) \end{cases}. \tag{42}$$

The location is represented by q_{j0}, and the position of the |state is represented by q_{j1}. A transformation of the solution space is needed to evaluate the benefits and drawbacks of the person's current location. The analogous parameter in the outcome space, T, can be represented as

$$\begin{bmatrix} x_{j0}^k \\ x_{j1}^k \end{bmatrix} = \begin{bmatrix} \frac{b_k - a_k}{2} & 0 \\ 0 & \frac{b_k - a_k}{2} \end{bmatrix} \begin{bmatrix} \cos\theta_{jk} \\ \sin\theta_{jk} \end{bmatrix} + \begin{bmatrix} \frac{b_k + a_k}{2} \\ \frac{b_k + a_k}{2} \end{bmatrix}. \tag{43}$$

As a result, either of the two optimization problem solutions may apply to a single member of individuals.

3.7.8 Implementation steps

The three phases of the QPMMCOA are described below.

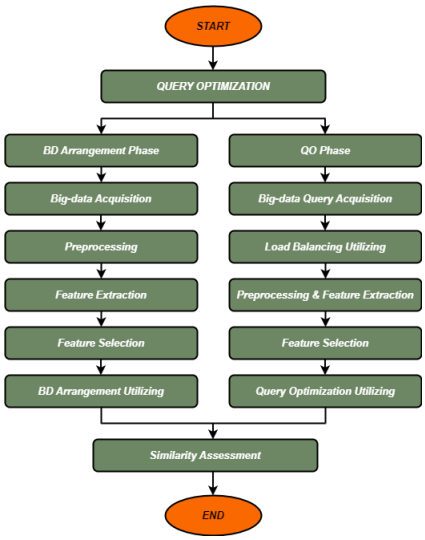


Figure 2: Flowchart

3.7.9 Rough search

The QPMMCOA arbitrarily explores for points m in the initial viable area coupled with f to produce the optimal values in fitness and then saves the initial set of u-optimized points when it begins the rough search. When determining the best value among the u points, a new viable area was established, and the starting viable area of the subsequent iteration process was considered during the rough search procedure. The primary goal of the rough search is to locate an approximate optimal solution inside the issue's viable zone. The determination approach of the first possible area ensures that the suggested approach is practical and avoids rapidly falling into the optimal location. Many optimization locations are used to build new feasible areas to expand the global search ability and improve the local growth capacity.

3.7.10 Precise search

The search procedure is much the same as in the rough search. One distinction between this approach and the rough search is the definition of the first viable areas. The ideal point was discovered during the "1" and "2" phases of the preceding layer's search. Precise search, on the other hand, establishes various searchable regions based on multiple optimization factors during the complete search procedure. Different viable zones prevent the precise search from focusing on a single location and entering the local optimum.
The parameters of the proposed optimization algorithm are population size, number of iterations, convergence criterion, mutation rate, crossover probability, and selection technique. The starting population size specifies the number of alternative solutions, while the number of iterations decides how many times the algorithm will be executed. Convergence criteria specify when the algorithm should end based on improved outcomes. The mutation rate and crossover probability control genetic diversity, allowing for the discovery of various solutions, whilst the selection method determines which solutions survive and propagate. The optimization problem aims to improve query processing efficiency in large data environments by reducing energy usage and computational costs while increasing accuracy and speed. The algorithm looks for the best settings to balance these competing goals and to provide precise query responses fast and effectively while using as few resources as possible. Figure 2 shows the flowchart.

4 Result and discussions

In this portion, the efficacy of the proposed work is assessed. The experimental results are shown below. The recommended approach processes differ from current traditional methods regarding specific quality measures. In the results section, further detail regarding how the theoretical frameworks previously addressed, such as Quantum parallel multi-layer Monte Carlo optimization (QPMMCOA) and Bayesian fuzzy clustering, were used will improve clarity and reproducibility. Providing specific numerical data, including performance indicators such as energy consumption reduction and query processing efficiency gains, will enable readers to reproduce the results effectively. Specifically,

highlighting specific benchmarks, comparison analyses with existing algorithms, and any statistical significance tests performed will help validate the proposed deep-learning-based query optimization strategy in big data contexts.

4.1 Experimental setup

The examination uses TensorFlow as a backdrop, using Python and KERAS on the Anaconda3 platform and utilizing a 16 GB RAM Intel i5 2.60 GHz processor running Windows 10.

4.2 Dataset description

We evaluated and analyzed our proposed approach using four standard datasets in the present investigation.
Dataset 1: IMDb dataset: Word vectors were created using the ACL Internet Movie Database (IMDb) dataset. Half (50,000) of the 100,000 text movie evaluations in the sample are test reviews without labels. To distinguish between positive and negative assessments, each of the remaining fifty thousand reviews is assigned a number ranging from 0 to 1. The labelled reviews are split in half to ensure a representative sample, with 12,500 good and 12,500 negative feedback in each set [50].
Dataset 2: Healthcare dataset: This dataset, which is meant for educational and non-commercial use, is made up of 10,000 records, every one of which represents a synthetic healthcare patient record with a variety of attributes, including patient demographics, health conditions, admission details, and more. It is entirely synthetic and does not contain real patient data [51].
Dataset 3: Twitter Dataset: Twitter data gathered from two American customer support profiles that provide English-language support to users in North America. These specialized Twitter accounts provide assistance and react to comments from users in real-time. Corporate support staff use Twitter to reply to these tweets. In our sample, 2 632 talks were recorded [52].
Dataset 4: Chicago crime data analysis: The dataset comprises six locations' hourly summertime measurements of the following parameters: wave height, water temperature, turbidity, beach name, measurement timestamp, transducer depth, and wave period [53].
An 80:20 split was used to separate the data into training and test sets, making use of 80% of the data for model training and 20% for testing. This division ensures that there is enough data for both performance assessment and model learning.

4.3 Evaluation metrics

Evaluation metrics are numerical measurements of a model's efficacy and effectiveness. These metrics aid in quantifying various performance factors, facilitating a more impartial and consistent comparison of multiple strategies. Evaluation criteria such as F1-Score, sensitivity, specificity, and accuracy are employed to assess the effectiveness of the suggested methodology. The control system block diagram is shown in figure 3. The explanations in depth are provided below.
Accuracy
Data must be derived from a reliable source and reflect the facts to be considered precise.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{44}$$

Sensitivity
The sensitivity of a collection of points of data is calculated as a percentage of the total number of couples of data points found. Cluster effectiveness and recall have a strong correlation.

$$Sensitivity = \frac{TP}{TP + FN}. \tag{45}$$

Specificity
The percentage of pairings of points of information that are appropriately assigned to the same cluster is known as specificity. It fluctuates directly with the efficiency with which new clusters develop.

$$Specificity = \frac{TP}{TP + FP}. \tag{46}$$

F1-Score
The calculated value of the F-measure is negatively correlated with accuracy and recollection; greater F-measures are

the outcome of higher precision and recall.

$$\frac{(2 \times precision \times recall)}{(precision + recall)}. \tag{47}$$

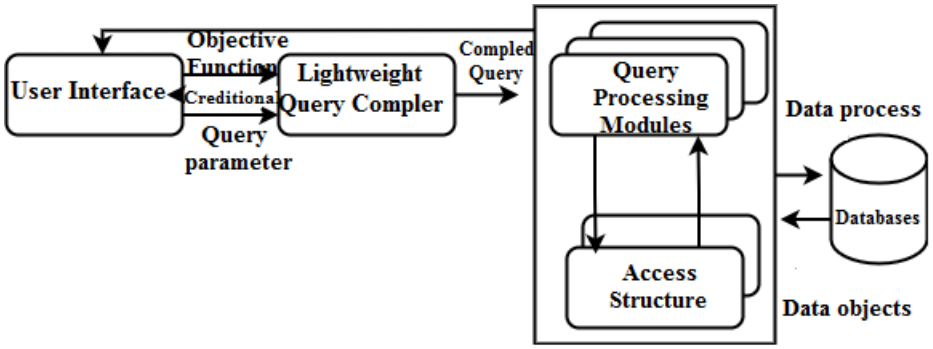


Figure 3: control system structure

Scenario 1 (Evaluation of Big data arrangement phase)

The big data arrangement stage is crucial to the performance of downstream analyses and applications and forms the basis of the entire data processing workflow. In this section, we evaluate the performance of the proposed approaches. A comparison of the performance of the BD phase is shown in Table 1.

Table 1: Performance comparison of big data arrangement phase

Datasets	Accuracy	Sensitivity	Specificity	F1-Score
IMDb	98.91	99.01	98.81	98.46
Chicago	98.67	98.76	99	98.86
Healthcare	99	98.63	99.04	98.92
Twitter	99.02	98.72	98.99	98.16

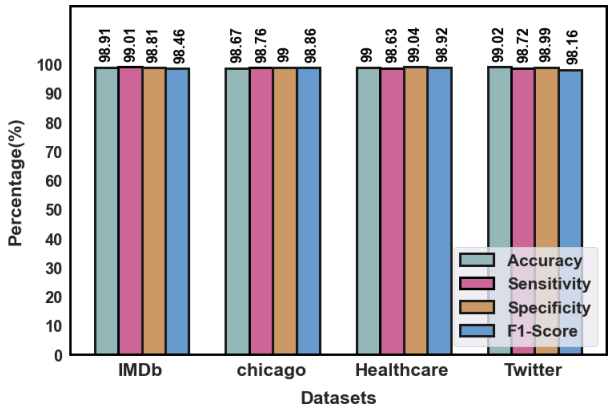


Figure 4: Differentiation of proposed BD phase

Metrics like accuracy, sensitivity, specificity, and F1-Score are employed to analyze the performance of the proposed approach. Figure 4 illustrates the differentiation of the proposed BD phase. The IMDb dataset obtains 98.91% accuracy, 99.01% sensitivity, 98.81% specificity, and 98.46% F1-Score. Chicago dataset yields 98.67% accuracy, 98.76% sensitivity, 99% specificity, and 98.86% F1-Score. The healthcare dataset yields 99% accuracy, 98.63% sensitivity, 99.04% specificity, and 98.92% F1-Score. Twitter dataset yields 99.02% accuracy, 98.72% sensitivity, 98.99% specificity,

and 98.16% of F1-Score.

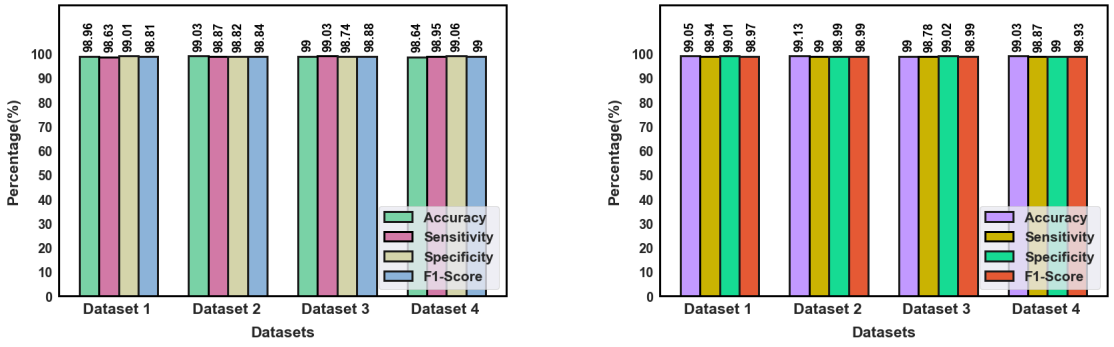
Scenario 2 (Evaluation of Query optimization phase)

A crucial phase in the data retrieval process is known as the "Query Optimization Phase" in handling databases. In contrast, the database management system works to improve query efficiency to guarantee more rapid and resource-efficient execution. This step, which includes choosing the best query execution plans from a range of options, is crucial to the functionality of database systems. A comparison of query optimization phases is shown in Table 2.

Table 2: Comparison of query optimization phases

Datasets	Load balancing phase				Query optimization phase			
	Acc (%)	Sen (%)	Spe (%)	F1 (%)	Acc (%)	Sen (%)	Spe (%)	F1 (%)
Dataset 1	98.96	98.63	99.01	98.81	99.05	98.94	99.01	98.97
Dataset 2	99.03	98.87	98.82	98.84	99.13	99	98.99	98.99
Dataset 3	99	99.03	98.74	98.88	99	98.78	99.02	98.99
Dataset 4	98.64	98.95	99.06	99	99.03	98.87	99	98.93

Figure 5: Performance comparison on 2nd phase (a) load balancing (b) QO phase



(a) (b)

Figure 5 shows the performance comparison on 2nd phase. We employed the QPMMCOA for query optimization. It yields greater performance than our proposed dataset.

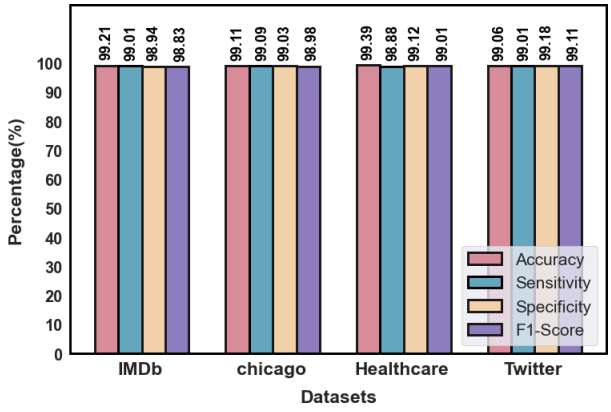
Scenario 3 (Evaluation of overall performances)

Organizations traversing the challenges of large-scale data processing use the assessment of overall performance in big data query optimization as a compass. It directs the improvement of optimization tactics, creating durable and adaptable systems to the constantly changing big data requirements and technology landscape, and being an expert at drawing valuable conclusions from enormous datasets. Overall performance differentiation is shown in Table 3.

Table 3: Overall Performance Comparison of Proposed Datasets

Datasets	Accuracy	Sensitivity	Specificity	F1-Score
IMDb	99.21	99.01	98.94	98.83
Chicago	99.11	99.09	99.03	98.98
Healthcare	99.39	98.88	99.12	99.01
Twitter	99.06	99.01	99.18	99.11

Figure 6: Differentiation of overall performances



The differentiation of overall performances is shown in Figure 6. The proposed approach yields more excellent accuracy, sensitivity, specificity, and F1-Score performances than existing research.

Table 4: Comparison of proposed approach vs. existing

Approach	**Datasets**	**Accuracy**	**Precision**	**Recall**
ACO-GA [27]	Hospital compare dataset	96.25%	95.47%	95.47%
CaM-BW[19]	Hospital compare dataset	96.98%	-	-
FCM[29]	Twitter data	90%	85.42%	85.27%
K-Means[14]	Twitter data, IMDb	93.21%	91.97%	-
NK-Means [3]	NYISO	96.11%	95.89%	93.47%
Proposed	IMDb, Chicago,Healthcare and Twitter	99.21%	99.02%	99.15%

Table 4 show the Differentiation of proposed vs existing approaches. The existing methods like K-means, NK-means, ACO-GA, FCM, and CaM-BW are employed to compare with our proposed approach. While differentiating from other methods, the presented approach yields a greater accuracy of 99.21%. The second-best performance is present in CaM-BW, which is 96.98%. (-) indicates the metrics not analyzed in their research.

Figure 7: Differentiation of proposed with existing approaches

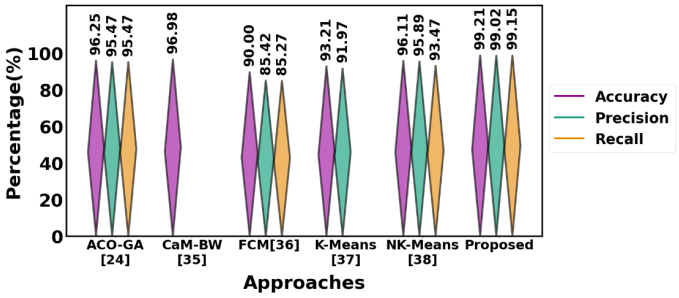
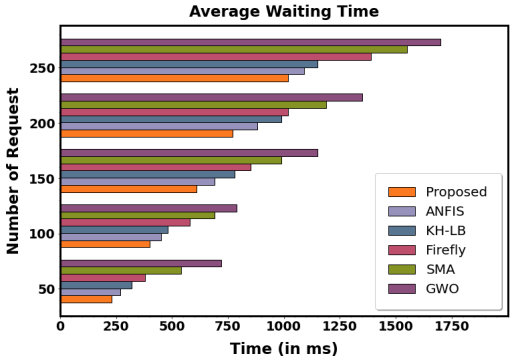
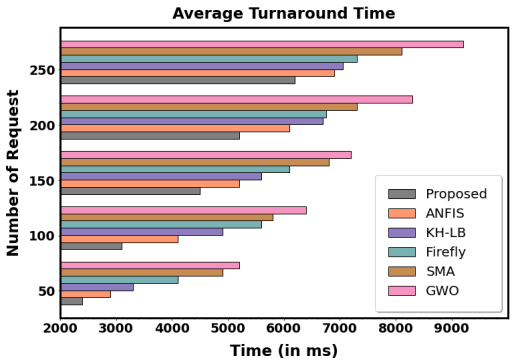


Figure 8: Comparison of average waiting time



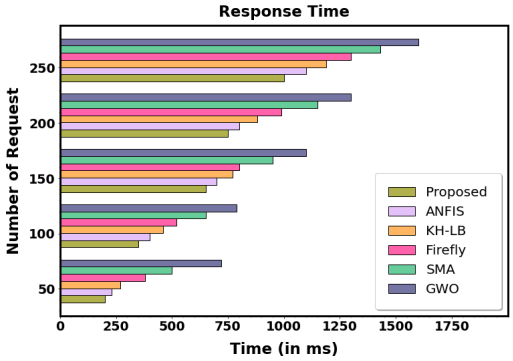
It follows that the proposed approach is very reliable and efficient. The completion speed is indicated by the system's LB time. The % measurement of LB is based on the time parameter. Figure 8 displays the effectiveness of the suggested and current LB strategies about LB time.

Figure 9: Comparison of average turnaround time



Based on how well they perform with LB time, Figure 9 compares the suggested and current LB strategies. The LB method is employed to improve system rapidity. This analysis indicates that the LB time grows as the number of requests expands and that the proposed takes less time for LB for all 50 to 250 requests. The turnaround period is the time needed to fulfil an order or finish a task. The proposed method takes less time to process a 50 request, which is faster than the other balancers for loads that are currently in use. Thus, it is demonstrated that the suggested balancer of loads is the one that operates with the most speed and efficiency. Similarly, the presented approach has a low response time compared to previous methods.

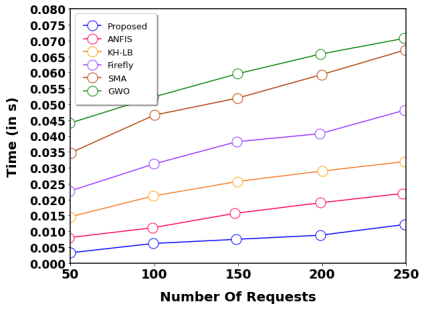
Figure 10: Comparison of response time



A comparison of response time is shown in Figure 10. Effectively, the suggested LB system assesses each data node's load. The proposed approach takes 0.003 s LB time for 50 requests from users, which is much less than the existing

ANFIS, KH-LB, firefly method, SMA, and Grey wolf optimization, which takes 0.012, 0.015 s, 0.023 s, 0.035 s, and 0.045 s, correspondingly. A system uses more energy when its time utilization rises, reducing efficiency. It is deduced that the suggested method is highly effective because it takes less time to complete LB for all 50 to 250 requests. Figure 11 illustrates the comparison of the balancing strategy.

Figure 11: Comparison of load balancing strategy



The results show a significant improvement in query optimization, notably in terms of energy consumption and accuracy when compared to previous algorithms. The beginning scenarios for these outcomes were a big dataset processed using the IDRSN method for feature extraction and the CVSA for feature selection. Furthermore, the QPMMCOA optimizer and Bayesian fuzzy clustering-based load balancer were calibrated using settings designed to balance query processing efficiency and energy consumption. The combination of these strategies meant that the system could handle the intricacies of massive data while optimizing queries in a resource-efficient way. The proposed approach has various advantages, including higher accuracy from sophisticated feature extraction and selection techniques, increased computational efficiency from algorithms such as CVSA and IDRSN, and enhanced efficiency from effective load balancing and query optimization. However, it has disadvantages. The suggested method has a few drawbacks: it can be challenging to implement. It may necessitate significant computational resources, which could be an issue for massive datasets or real-time applications.

Table 5: Differentiation of proposed with existing approaches using proposed datasets

Dataset	Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Execution Time (sec)
IMDb Dataset	DeepConvNet [40]	91.8	90.1	89.2	89.6	3.0
	SVM [40]	89.5	85.0	87.0	86.0	2.2
	Proposed	99.21	99.01	98.94	98.83	2.5
Healthcare dataset	XGBoost [18]	94.5	93.5	94.0	93.7	3.5
	Random Forest [41]	92.1	90.5	91.0	90.7	4.0
	Proposed Method	99.39	98.88	99.12	99.01	3.2
Twitter Dataset	Proposed	99.06	99.01	99.18	99.11	1.8
	LSTM-based [19]	84.5	80.2	82.0	81.1	2.0
	BERT-based [21]	87.0	85.5	86.0	85.7	2.5
Chicago Dataset	Proposed Method	99.11	99.09	99.03	98.98	4.2
	CNN [12]	91.1	89.3	90.0	89.6	4.5
	KNN [6]	89.0	87.4	88.0	87.7	3.8

The performance of the proposed strategy and existing techniques are contrasted in table 5 using four datasets: Chicago Crime, Twitter, Healthcare, and IMDb. For several approaches, key performance measures are shown, such as execution time, F1-score, recall, accuracy, and precision. The proposed method has competitive execution times and continuously exceeds current models in terms of accuracy, precision, recall, and F1 score across all datasets. With accuracy scores surpassing 99% for the Healthcare, Twitter, and Chicago datasets and 99.21% for the IMDb dataset, the proposed strategy notably performs exceptionally well while preserving quick execution times.

Table 6: Overall performance differentiation of proposed with existing approaches

Technique	Accuracy (%)	Sensitivity (%)	Specificity (%)	F1-Score (%)	Execution Time (sec)
Indexing-based Optimization [8]	95.5	92.3	98.1	94.6	10.2
Query Caching [29]	97.3	96.5	98.2	97.3	5.8
Copart [30]	94.8	93.2	97.9	95.5	12.3
Cost-based Optimization [42]	96.2	95.1	97.0	96.0	8.4
Machine Learning-based Optimization [44]	99.1	98.7	99.2	98.9	7.5
Distributed Query Optimization [31]	98.5	97.4	98.8	98.1	6.2
Approximation-based Optimization [50]	93.7	90.5	96.8	93.6	4.1
Proposed	99.21	99.02	99.15	99.07	3.2

A comparison of different query optimization strategies is shown in Table 6 according to essential performance indicators such as execution time, F1-score, accuracy, sensitivity, and specificity. The findings demonstrate the excellent accuracy and efficiency attained using distributed query optimization methods and machine learning-based approaches. While keeping a short execution time (3.2 seconds), the proposed approach achieves the highest accuracy (99.21%), sensitivity (99.02%), specificity (99.15%), and F1-score (99.07%), outperforming the existing methods. This illustrates the way the proposed strategy optimizes searches with increased accuracy and lower computing costs.

Conclusion and future scope

Optimizing big data queries is essential to organizing and extracting insight from significant, complicated information. Optimizing queries becomes crucial in big data, where datasets can be enormous and dispersed across clusters, to guarantee timely and effective data processing. One of the current challenges in big data query optimization is managing a variety of data formats and structures, which might result in query performance that could be at its best. Furthermore, scalability problems caused by the sheer volume of data make it challenging to reliably provide real-time query responsiveness in enormously big and dynamic datasets. We introduced a novel deep learning-based big data arrangement and fuzzy-based load balancing approaches to overcome the existing limitations. Our proposed research consists of two phases: the big data arrangement phase and the query optimization phase. In the first phase, we utilized an improved deep residual shrinkage network to arrange the big data. The essential features like lift, support, confidence, and closed frequent item set are extracted. Then, the vital features are extracted using the Chaotic Vertex Search algorithm (CVSA).

Load balancing is used in big data query optimization to achieve an effective and fair allocation of computing resources and increase efficiency, scalability, fault tolerance, and flexibility to changing workloads within a big data processing environment. We utilized a Bayesian fuzzy clustering algorithm to balance the load in the second phase. Then, the queries are optimized by employing a QPMMCOA. The suggested method's output is compared to present techniques during performance evaluation. The results showed that the suggested optimizer provides higher accuracy compared to the current methods. The suggested optimizer took 0.008 s (low) LB time, achieved over 99% accuracy, and had a 9.2 s (low) response time. The investigation demonstrated that, compared to the earlier QO strategies, the suggested approach is more accurate, stable, and efficient. Future efforts will concentrate on combining comparable queries to reduce the time-consuming nature of the query optimization step. Future developments in data mining techniques will be anticipated to increase system effectiveness.

Declaration

Ethics approval: This material is the authors' own original work, which has not been previously published elsewhere. The paper reflects the authors' own research and analysis in a truthful and complete manner.

Competing interests: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

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

Funding: No funding was received to assist with the preparation of this manuscript.

Availability of data and material: Data will be available when requested.

Authors' contributions: The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

References

[1] H. B. Abdalla, A. M. Ahmed, M. A. Al Sibahee, *Optimization-driven mapreduce framework for indexing and retrieval of big data*, KSII Transactions on Internet and Information Systems (TIIS), **14**(5) (2020), 1886-1908. http://doi.org/10.3837/tiis.2020.05.002

[2] L. Abualigah, A. H. Gandomi, M. A. Elaziz, H. A. Hamad, M. Omari, M. Alshinwan, A. M. Khasawneh, *Advances in meta-heuristic optimization algorithms in big data text clustering*, Electronics, **10**(2) (2021), 101. https://doi.org/10.3390/electronics10020101

[3] R. Akram, N. Ayub, I. Khan, F. R. Albogamy, G. Rukh, S. Khan, K. Rizwan, *Towards big data electricity theft detection based on improved rusboost classifiers in smart grid*, Energies, **14**(23) (2021), 8029. https://doi.org/10.3390/en14238029

[4] M. Q. Bashabsheh, L. Abualigah, M. Alshinwan, *Big data analysis using hybrid meta-heuristic optimization algorithm and MapReduce framework, in integrating meta-heuristics and machine learning for real-world optimization problems*, Cham: Springer International Publishing, (2022), 181-223. https://doi.org/10.1007/978-3-030-99079-4_8

[5] J. Bater, Y. Park, X. He, X. Wang, J. Rogers, *Saqe: Practical privacy-preserving approximate query processing for data federations*, Proceedings of the VLDB Endowment, **13**(12) (2020), 2691-2705. https://doi.org/10.14778/3407790.3407854

[6] R. Chi, H. Li, D. Shen, Z. Hou, B. Huang, *Enhanced P-type control: Indirect adaptive learning from set-point updates*, IEEE Transactions on Automatic Control, **68**(3) (2022), 1600-1613. https://doi.org/10.1109/TAC.2022.3154347

[7] D. Choi, J. Wee, S. Song, H. Lee, J. Lim, K. Bok, J. Yoo, *K-NN query optimization for high-dimensional index using machine learning*, Electronics, **12**(11) (2023), 2375. https://doi.org/10.3390/electronics12112375

[8] Q. T. Doan, A. S. M. Kayes, W. Rahayu, K. Nguyen, *A framework for IoT streaming data indexing and query optimization*, IEEE Sensors Journal, **22**(14) (2022), 14436-14447. https://doi.org/10.1109/JSEN.2022.3149901

[9] K. Dubey, A. Kumar, R. Agrawal, *An efficient ACO-PSO-based framework for data classification and preprocessing in big data*, Evolutionary Intelligence, **14** (2021), 909-922. https://doi.org/10.1007/s12065-020-00477-7

[10] W. Ge, X. Li, C. Yuan, Y. Huang, *Correlation-aware partitioning for skewed range query optimization*, World Wide Web, **22**(1) (2019), 125-151. https://doi.org/10.1007/s11280-018-0547-4

[11] D. Geng, C. Zhang, C. Xia, X. Xia, Q. Liu, X. Fu, *Big data-based improved data acquisition and storage system for designing industrial data platform*, IEEE Access, **7** (2019), 44574-44582. https://doi.org/10.1109/ACCESS.2019.2909060

[12] 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, Springer Singapore, **1** (2022), 307-318. https://doi.org/10.1007/978-981-16-2164-2_25

[13] Y. Guo, Z. Shao, *Cymo: A storage model with query-aware indexing for spatio-temporal big data*, In 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), (2022), 122-132. https://doi.org/10.1109/ICDCS54860.2022.00021

[14] H. Hu, J. Liu, X. Zhang, M. Fang, *An effective and adaptable K-means algorithm for big data cluster analysis*, Pattern Recognition, **139** (2023), 109404. https://doi.org/10.1016/j.patcog.2023.109404

[15] M. Jagdish, N. Anand, K. Gaurav, S. Baseer, A. Alqahtani, V. Saravanan, *Multihoming big data network using blockchain-based query optimization scheme*, Wireless Communications and Mobile Computing, **1** (2022), 1-12. https://doi.org/10.1155/2022/7768169

[16] N. I. N. G. Jing, *Neural network-based pattern recognition in the framework of edge computing*, Science and Technology, **27**(1) (2024), 106-119.

[17] H. Kour, M. K. Gupta, *Hybrid evolutionary intelligent network for sentiment analysis using twitter data during COVID-19 pandemic*, Expert Systems, **41**(3) (2024), e13489. https://doi.org/10.1111/exsy.13489

[18] D. Kumar, V. K. Jha, *An improved query optimization process in big data using ACO-GA algorithm and HDFS map reduce technique*, Distributed and Parallel Databases, **39** (2021), 79-96. https://doi.org/10.1007/s10619-020-07285-z

[19] D. Kumar, V. K. Jha, *An efficient query optimization technique in big data using σ-ANFIS load balancer and CaM-BW optimizer*, The Journal of Supercomputing, **77**(11) (2021), 13018-13045. https://doi.org/10.1007/s11227-021-03793-6

[20] R. Kumar, P. Kumar, Y. Kumar, *Integrating big data driven sentiments polarity and ABC-optimized LSTM for time series forecasting*, Multimedia Tools and Applications, **81**(24) (2022), 34595-34614. https://doi.org/10.1007/s11042-021-11029-1

[21] V. N. Kumar, A. Kumar P. S., *An efficient and scalable SPARQL query processing framework for big data using MapReduce and hybrid optimum load balancing*, Data and Knowledge Engineering, **148**(1) (2023), 102239. https://doi.org/10.1016/j.datak.2023.102239

[22] D. Li, L. Deng, Z. Cai, *Statistical analysis of tourist flow in tourist spots based on big data platform and DA-HKRVM algorithms*, Personal and Ubiquitous Computing, **24** (2020), 87-101. https://doi.org/10.1007/s00779-019-01341-x

[23] X. Li, H. Liu, W. Wang, Y. Zheng, H. Lv, Z. Lv, *Big data analysis of the internet of things in the digital twins of smart city based on deep learning*, Future Generation Computer Systems, **128** (2022), 167-177. https://doi.org/10.1016/j.future.2021.10.006

[24] D. Mahajan, C. Blakeney, Z. Zong, *Improving the energy efficiency of relational and NoSQL databases via query optimizations*, Sustainable Computing: Informatics and Systems, **22**(1) (2019), 120-133. https://doi.org/10.1016/j.suscom.2019.01.017

[25] G. Manogaran, P. M. Shakeel, S. Baskar, C. H. Hsu, S. N. Kadry, R. Sundarasekar, B. A. Muthu, *FDM: Fuzzy-optimized data management technique for improving big data analytics*, IEEE Transactions on Fuzzy Systems, **29**(1) (2020), 177-185. https://doi.org/10.1109/TFUZZ.2020.3016346

[26] S. Meera, C. Sundar, *A hybrid metaheuristic approach for efficient feature selection methods in big data*, Journal of Ambient Intelligence and Humanized Computing, **12** (2021), 3743-3751. https://doi.org/10.1007/s12652-019-01656-w

[27] P. Michiardi, D. Carra, S. Migliorini, *Cache-based multi-query optimization for data-intensive scalable computing frameworks*, Information Systems Frontiers, **23**(1) (2021), 35-51. https://doi.org/10.1007/s10796-020-09995-2

[28] S. Migliorini, A. Belussi, E. Quintarelli, D. Carra, *CoPart: A context-based partitioning technique for big data*, Journal of Big Data, **8** (2021), 1-28. https://doi.org/10.1186/s40537-021-00410-4

[29] A. Murugan, D. Gobinath, S. G. Kumar, B. Muruganantham, S. Velusamy, *A time efficient and accurate retrieval of range aggregate queries using fuzzy clustering means (FCM) approach*, International Journal of Electrical and Computer Engineering, **10**(1) (2020), 415. https://doi.org/10.11591/ijece.v10i1.pp415-420

[30] N. Orensa, *A design framework for efficient distributed analytics on structured big data*, Doctoral Dissertation, University of Saskatchewan, 2021.

[31] N. G. Praveena, S. S. Nath, *A fuzzy based efficient and blockchain oriented secured routing in vehicular Ad-Hoc networks*, Iranian Journal of Fuzzy Systems, **21**(6) (2024), 15-31.

[32] M. M. Rahman, S. Islam, M. Kamruzzaman, Z. H. Joy, *Advanced query optimization in SQL databases for real-time big data analytics*, Academic Journal on Business Administration, Innovation and Sustainability, **4**(3) (2024), 1-14. https://doi.org/10.1109/access.2022.3141589

[33] V. Ravuri, S. Vasundra, *Moth-flame optimization-bat optimization: Map-reduce framework for big data clustering using the Moth-flame bat optimization and sparse fuzzy C-means*, Big Data, **8**(3) (2020), 203-217. https://doi.org/10.1089/big.2019.0125

[34] R. C. Roman, R. E. Precup, E. M. Petriu, A. I. Borlea, *Hybrid data-driven active disturbance rejection sliding mode control with tower crane systems validation*, Science and Technology, **27** (2024), 3-17.

[35] R. Sahal, M. H. Khafagy, F. A. Omara, *Exploiting coarse-grained reused-based opportunities in big data multi-query optimization*, Journal of Computational Science, **26** (2018), 432-452. https://doi.org/10.1016/j.jocs.2017.05.023

[36] R. Sahal, M. Nihad, M. H. Khafagy, F. A. Omara, *iHOME: Index-based JOIN query optimization for limited big data storage*, Journal of Grid Computing, **16** (2018), 345-380. https://doi.org/10.1007/s10723-018-9431-9

[37] M. Sharma, G. Singh, R. Singh, *Clinical decision support system query optimizer using hybrid firefly and controlled genetic algorithm*, Journal of King Saud University-Computer and Information Sciences, **33**(7) (2021), 798-809. https://doi.org/10.1016/j.jksuci.2018.06.007

[38] T. Siddiqui, A. Jindal, S. Qiao, H. Patel, W. Le, *Cost models for big data query processing: Learning, retrofitting, and our findings*, In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, (2020), 99-113. https://doi.org/10.1145/3318464.3380584

[39] D. Sujatha, M. Subramaniam, C. R. Rene Robin, *A new design of multimedia big data retrieval enabled by deep feature learning and adaptive semantic similarity function*, Multimedia Systems, **28**(3) (2022), 1039-1058. https://doi.org/10.1007/s00530-022-00897-8

[40] M. Sun, L. Sun, *Optimization of artificial intelligence in localized big data real-time query processing task scheduling algorithm*, Frontiers in Physics, **12** (2024), 1484115. https://doi.org/10.3389/fphy.2024.1484115

[41] M. R. Sundarakumar, D. Salangai Nayagi, V. Vinodhini, S. VinayagaPriya, M. Marimuthu, S. Basheer, J. A. Renoald, *A heuristic approach to improve the data processing in big data using enhanced Salp Swarm algorithm (ESSA) and MK-means algorithm*, Journal of Intelligent and Fuzzy Systems, **45**(2) (2023), 2625-2640. https://doi.org/10.3233/JIFS-231389

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

[43] D. R. Thirupurasundari, R. Kumar, H. K. Palani, S. Ilangovan, P. G. Senthilvel, *Optimizing query performance in big data systems using machine learning algorithms*, In 2023 International Conference on Communication, Security and Artificial Intelligence (ICCSAI), (2023), 891-895. https://doi.org/10.1109/ICCSAI59793.2023.10421253

[44] W. Wang, H. Guo, X. Li, S. Tang, J. Xia, Z. Lv, *Deep learning for assessment of environmental satisfaction using BIM big data in energy efficient building digital twins*, Sustainable Energy Technologies and Assessments, **50** (2022), 101897. https://doi.org/10.1016/j.seta.2021.101897

[45] C. Xu, X. Du, Z. Yan, X. Fan, *ScienceEarth: A big data platform for remote sensing data processing*, Remote Sensing, **12**(4) (2020), 607. https://doi.org/10.3390/rs12040607

[46] J. Yang, C. Zhao, C. Xing, *Big data market optimization pricing model based on data quality*, Complexity, **1** (2019), 1-13. https://doi.org/10.1155/2019/5964068

[47] M. Zhang, Y. Chen, W. Susilo, *PPO-CPQ: A privacy-preserving optimization of clinical pathway query for e-healthcare systems*, IEEE Internet of Things Journal, **7**(10) (2020), 10660-10672. https://doi.org/10.1109/JIOT.2020.3007518

[48] P. Zhang, S. Cui, B. Du, *Fuzzy portfolio selection with different risk attitudes based on machine learning*, Iranian Journal of Fuzzy Systems, **22**(1) (2025), 1-21. https://doi.org/10.22111/ijfs.2025.47341.8338

[49] W. Zhang, T. Leng, H. Sun, *Optimization research of spatial big data approximate query algorithm in the context of smart city*, In International Conference on Smart Applications and Sustainability in the Artificial Intelligence of Things, Cham: Springer Nature Switzerland, (2024), 737-745. https://doi.org/10.1007

[50] https://www.kaggle.com/datasets/noordeen/big-city-health-data

[51] https://github.com/LearnDataSci/articles/blob/master/Python

[52] https://www.kaggle.com/datasets/cms/hospital-general-information

[53] https://www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter