International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Performance Analysis of Meta-Heuristic-Based Query Optimization Algorithms for Large-scale Decision Support Systems

Anita Mohanty[1*], Sambit Kumar Mishra[2]

Submitted: 28/01/2024 **Revised**: 06/03/2024 **Accepted**: 14/03/2024

Abstract: Database systems continue to be fascinated in the alluring quest of query optimisation, a field distinguished by continual heuristic developments. Swift data access and analysis are of utmost importance in the dynamic world of Decision Support System (DSS) databases. This research introduces a novel stochastic DSS query optimizer, expanding the capabilities of existing genetic approaches. The Enumeration-based Query Optimizer (EBQO), Genetic-based Query Optimizer (GBQO), and the recently suggested Stochastic-based Query Optimizer (SBQO) emerge as prominent candidates within the spectrum of query optimisation approaches. In comparison to EBQO and GBQO, SBQO, a stochastic technique, exhibits superior relevance to query optimisation. Notably, SBQO surpasses its rivals in two essential areas: runtime effectiveness and Total Costs Optimisation. This notable efficiency underlines how well stochastic strategies work for obtaining the best results when it comes to query optimisation. The importance of stochastic approaches in improving query optimisation efforts is highlighted by these results, opening up a path to more effective and favourable outcomes. This study provides a convincing demonstration of the significant advantages that cutting-edge stochastic techniques, such as SBQO, may bring to the field of query optimisation, a crucial component of effective database administration and decision support.

Keywords: *Cost Analysis, Genetic Algorithm, Meta-heuristic Algorithms, Performance Evaluation, Query Optimization, Run-time Analysis, Stochastic Approach.*

1. Introduction

A database is an amalgamation of nearly linked data that is put together in a way that meets the information needs of an enterprise. It is a place where individuals can store and view data that they can share with each other. It has information about how it works and a complete account of it. In the past few decades, computer technology has made giant leaps forward. The way that many companies operate and handle data has changed in a big way.

Additionally, there has been a big jump in the number of people who use databases and the organizations that use them. A database system is a group of programs that work together to build, store, retrieve, and manage information in the database. Before, the whole database was supposed to be put on a computer machine and shared by everyone who used the database. People knew this database management method was called "centralized database management." This method solved a lot of the problems that standard file-based systems possessed, such as duplicate data, no sharing, security issues, inconsistency, etc. Afterward, it was seriously thought that putting the whole database on a single site was one of the biggest problems with this method because it slowed it down. Also, this method did not make "Access Time" and "Response Time" faster as the size of the database grew to significantly higher levels. In the 1980s, when database systems and computer networking came together, a new name emerged: "distributed database system." It was a significant change in the way computer technology worked. The distributed database system fixed some of the problems with standard database systems.

Query is a sentence or group of statements that do basic database tasks like "read," "write," "delete," and "update" correctly. It is an essential part of managing and getting info. Most of the time, distributed queries are more complicated than centralized queries. Online Transaction Processing (OLTP) and Decision Support System (DSS) queries are more spread queries. DSS question is spread out in the world. In general, it is hard to do and takes longer to do. With these queries, you can get info from both nearby and faraway sites.

In contrast to OLTP searches, these queries usually deal with a large amount of data. DSS searches use a lot of I/O, processing and communication resources, and they can cause a distributed database system's CPU or even memory server to stop working suddenly. Also, the average running time of a spread DSS query is hard to predict. DSS queries work on relationships with sizes of mega bytes, giga bytes, or even bigger. In the end, query optimization was the most challenging problem for database sexperts. In recent years,

[1]*Research Scholar(Comp.sc & Engg), Biju Patnaik University of Technology, Rourkela ,Odisha; and*
[1] *Department of MCA, Ajay Binay Institute of Technology, Cuttack*
ORCID ID : 0009-0005-9220-379X
[2] *Department of Computer Sc.&Engg., Gandhi Institute for Education and Technology, Baniatangi*
ORCID ID : 0000-0002-5767-6787
** Corresponding Author Email: anitamohanty56@gmail.com*

much attention has been paid to query efficiency in distributed database systems. It is a way to figure out the best way to run a query regarding Total Costs or Response Time. If a query's processing plan isn't the best, it will either use too many system resources or take too long to run. The main goal of this study is to look at how well different stochastic distributed DSS query optimizers work. The outcomes are contrasted based on how long the distributed DSS query took to run and how many system resources it used. Here, a new stochastic query optimizer, namely, the Stochastic-based Query Optimizer (SBQO) is suggested to optimize distributed DSS queries. The results for SBQO are compared to GBQO and EBQO algorithms for different set of queries corresponding to performance factors like total cost, run-time, and percentage of reduction in cost.

The paper is broken up into different parts. In Section 2 of the paper, the linked work was talked about. In Section 3, the problem statement is set out. Section 4 explains what query optimization is all about. In Section 5, the designs of different query optimizers along with proposed SBQO algorithm is presented. In Section 6, we talked about the cost coefficient method and how to set up an experiment along with the discussion and the findings. In Section 7, the conclusion has been made and future scope of the study is discussed.

2. Related Work

Yao and Hevner are the ones who came up with query optimization. In the late 1970s, authors optimized queries using a heuristic with an exhaustive enumeration method. In the 1980s, different query optimization strategies were suggested by Ceri and Palagatti, Chen and Li, Yu and Chang, Peter Apers, Lam, and Martin, and other critical researchers. In 1995, Rho and March added more to the query optimization model. In the 21st century, researchers like Ahmat Cosar and Zehai Zhou used a method called "Genetic Algorithm" to improve the performance of spread queries [1, 5, 29]. In the past, most searches were optimized using "Exhaustive Enumeration" with some heuristics algorithms (such as Dynamic Programming, Branch and Bound, Greedy Algorithm, etc.).

But this method wasn't suitable for big, complicated queries because it rarely came up with the best query placement plan in a set amount of time. For a complex question, it took minutes, hours, or even days to develop the best plan for running the query [6,7]. In randomized optimization methods, a set of random moves was used to find the best answer.

Every search space answer was shown as a solution point. As an edge, a connection between two answer points was made by making random moves. The set of unexpected moves depends greatly on how the optimization problem is set up and the answers. Some examples of randomized optimization methods are "Iterative Improvements," "Simulated Annealing," "Random Sampling," etc. In recent years, evolutionary processes have been used to improve the performance of spread queries. The idea behind evolutionary techniques is that a group changes over time. Some of the essential things about evolutionary methods are that they can deal with imperfection, uncertainty, and only some of the truth to make things easy to understand, strong, cheap to solve, and more in line with reality. Some popular evolutionary methods are "Genetic Algorithms," "Swarm Intelligence," "Memetic Algorithms," "ACO," and so on [8–11].

3. Problem Formulation

An NP-Hard task is optimizing queries. Several heuristics have been implemented recently; these offer novel strategies for improving the query processing loop. We're still on the lookout for more optimal answers. Optimizing a query typically entails moving subqueries around, rearranging the query's structure, or effectively allocating sub-operations to various sites (operation site allocation). The distribution of places for operations is one of the most studied issues in distributed databases. As a result, the key objective of this study has been to delegate sub-operations to various locations of a distributed database network to optimize a search query. Gigabytes, petabytes, and more, decision support system queries interpret enormous amounts of data. This category of inquiries is exempt from the response time. However, a key issue is the system resources needed to run the query. To address the challenge of where to run distributed DSS queries, an optimizer has been developed. A 'SQL' based decision support system query is first broken down into relational algebra expressions (sub-operations) based on selection,' 'projection,' 'join,' and semi-join to find an ideal operation site allocation plan. By experimenting with different hybrids of operations and locations, these sub-operations are then assigned to other places for execution. Each sub-operations cost is calculated by factoring in the size of the relation/fragment involved in the query, the site allotted, and the values of the input-output, processing, and communication costs coefficients. Figure 1 depicts the operation site allocation difficulty.

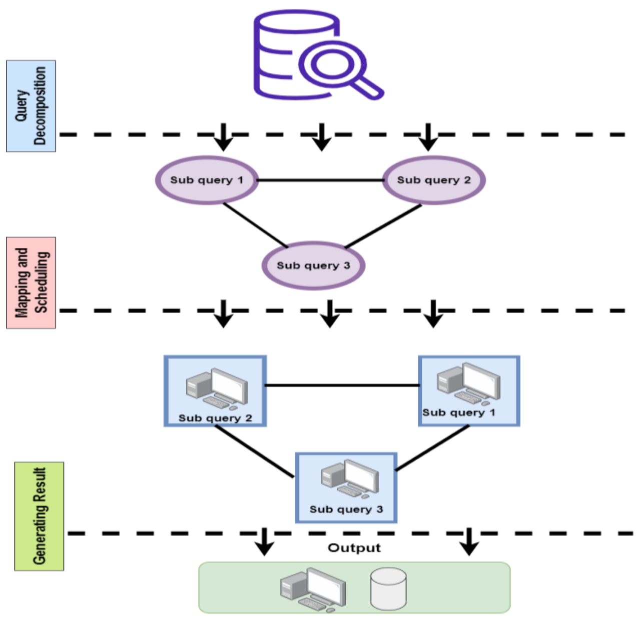

Fig 1. Proposed framework for query processing.

In this work, we present an optimization of a DSS query that uses comprehensive sampling, chaotic, restricted stochastic, and entropy-based restricted stochastic methods. Additionally, the impact of data replication on the tuning of distributed queries has been studied. In addition, the relationship between the quantity of join operations and the 'Total Costs' of the distributed DSS query is investigated. At long last, a statistical study of DSS query optimizer consistency has been completed.

4. DSS Query Optimization

Numerous operation site allocation plans to carry out the query are generated by the process known as query optimization. Selecting a more effective query execution plan that lowers the Total Costs of the distributed decision support system query is the goal of the operation site allocation problem. Typically, a software component known as a professional database management system uses a query optimizer to automate the query optimization process. Cost Model, Search Space, and Search Strategy are the three parts that make up a query optimizer [2,12]. The search space represents alternative query execution plans. Different execution plans generated by a query optimization approach are evaluated based on a query's 'Total Costs' to determine the best potential operation site allocation strategy. The cost model determines the 'Costs' for each query execution plan. A query's costs are calculated based on its operation and execution environment. Frequently, an 'objective function' or 'Costs Function' will be used to indicate the 'Costs' involved in a given situation. It is often built based on how long a query takes or how much system memory it consumes. The size and cardinality of relations, the number of blocks, and device input/output speeds all play significant roles. The job of the search strategy is to explore the search space and locate the optimal query execution plan [13].

5. Query Optimization using Meta-heuristic Approaches

It is clear from previous research results that when using exhaustive enumeration approaches, scaling up NP-hard problems results in a situation that is almost intractable. However, by utilising stochastic techniques, this situation can be effectively resolved. The Genetic Algorithm (GA) stands out as a strong contender among these. The dimensions of the search space remain unrelated to the computational time needed to reach a solution inside the GA framework. Because of this inherent trait, the evolutionary technique is ideally suited for query optimisation in distributed computing environments [8, 14, 15, 10, 16].

The conceptual origins of the "Genetic Algorithm," sometimes known as "GA," can be traced to John Holland's groundbreaking work. These algorithms make up a group of search tactics that have been painstakingly designed to closely resemble the basic ideas behind evolution's natural biological process. The term "GA" derives its meaning from the profound analogies it makes to the complex field of genetics. In more technical terms, genetic algorithms function as stochastic methods capable of delivering solutions of excellent quality while preserving a low degree of temporal complexity. This is made possible by coordinating a population of several individual chromosomes, which iteratively evolve under clear selection criteria and ultimately reach a state that best fulfils the desired function. These algorithmic paradigms generate efficacy through the collective effort of solution populations, in contrast to conventional techniques that concentrate on solo solutions. A set of heuristic processes, including selection, crossover, and mutation, which are all coordinated to improve solution quality, serves as the foundation for this endeavour [16, 8].

The adaptability of "Genetic Algorithms" is demonstrated by the wide range of issue fields in which they can be applied. Their effectiveness has been maximised in numerous fields, including image processing, environmental sciences, time series analysis, task scheduling, bioinformatics, clustering, game theory, artificial intelligence, and aviation [17–22]. These algorithms fundamentally represent the search for superior solutions among a variety of potential options. The GA approach, as stated, departs from the typical style of operating from a single solution by starting its optimisation project with a group of solutions. Through random generation, this initial solution population is created. Each solution to a problem is represented in this method by a chromosome, which is an encoded string made up of binary bits or characters (genes). A fitness value that measures the effectiveness of the treatment is augmenting the chromosomal composition. The population is made up of the collection of chromosomes and their related fitness ratings. This collection represents a certain generation at any given time. The fitness function, a crucial variable that crystallises the problem's optimisation objective, is at the heart of GA's effectiveness.

A key component of GA's mechanism is the reproduction of children, which is accomplished by pairing chromosomes according to the magnitudes of their fitness. This relationship triggers a crossover operation, in which the genetic characteristics of both parent chromosomes combine to produce children with superior qualities. Following crossover, the genetic characteristics of the progeny are further altered, a transformation known as mutation. The created progeny can be given improved efficacy through mutation. Until the specified termination requirements are satisfied, the algorithm's iterative advancement continues [16, 23].

5.1. Enumeration-Based Query Optimization (EBQO)

A deterministically driven strategy that conducts a thorough investigation of the solution space is the exhaustive enumeration approach. This approach ensures a thorough assessment of prospective solutions by methodically generating and examining all conceivable combinations inside the search space. Its conceptual framework's simplicity makes it easily understandable and implementable. However, it loses effectiveness when dealing with complex and wide-ranging issue domains of significant size.

For instance, the Enumeration approach meticulously examines every possible arrangement of query execution plans when used to solve the complexity inherent in a DSS Operation Site Allocation problem. However, cases with significant problem dimensions or complex structures limit the applicability of the method. The Enumeration-based Query Optimizer (EBQO) was developed with the goal of resolving the aforementioned research challenge in a remote database environment. It is based on a set of discerning decision variables. The process underpinning the design of EBQO is informed by all of these decision variables, allowing for a thorough and organised approach to dealing with the complex intricacies of the distributed database landscape.

5.2. Genetic-based Query Optimization (GBQO)

A novel strategy known as the Genetic-based Query Optimizer (GBQO) has been meticulously developed to handle the complexities of the operation site allocation problem within distributed Decision Support System (DSS) queries, building on the theoretical framework established by Rho and March [14]. With a deliberately produced beginning population that is stochastic, the GBQO approach starts its optimisation trip. The development of a chromosome, the blueprint of which is intricately woven from factors including the total number of operations and the total number of sites implicated in the query scenario, is a key aspect of GBQO's design. It is noteworthy that the chromosome's architecture is so carefully designed that its length is one unit less than the total number of operations contained in a given query [14].

It is important to emphasise the key assumption that guides the GBQO operating framework. This method offers a smooth link between well-established theoretical foundations and the practical requirements of distributed DSS query optimisation in the real world since it is strongly rooted in the ideas outlined by Rho and March [14]. A pseudo-code representation of the key operational sequences contained inside GBQO has been painstakingly developed to concretize this operational paradigm. This pseudo-code serves as a thorough manual, outlining the complex sequence of steps that make up the GBQO optimisation procedure. By doing this, it not only provides a thorough and organised understanding of the method but also serves as a springboard for further investigation and improvement in the field of distributed query optimisation strategies.

5.3. Stochastic-based Query Optimization (SBQO)

The earlier discussed GBQO technique, starts its optimisation strategy by creating an initial population using a stochastic process. At its core, it manages the distribution of sub-operations for a given DSS query within a distributed network context. The distinctive feature of GBQO's innovation is its chromosomal design, which is characterised by a carefully restrained expansion. According to this novel design idea, projection sub-procedures must always be assigned to run on the same computing node as their related selection operations. The chromosomal design's strategic configuration results in a noticeable decrease in the "Processing Costs" connected with the query, which has the knock-on effect of lowering the "Total Costs" for the encompassing DSS inquiry. The unique character of the chromosomal design of GBQO resonates as a trailblazing departure from traditional genetic algorithm approaches. A paradigm shift may be seen in the instruction to closely align projection sub-procedures with the location of their parent selection operations. This customised design not only increases the overall optimisation efficacy but also adds a novel query optimisation strategy dimension. The addition of the essential genetic algorithm operators "Selection," "Crossover," and "Mutation" is a key aspect of GBQO's adaption. These operators undergo careful alteration to cooperatively resonate with the distinctive features of the chromosomal design.

However, GBQO does not provide a 100% guarantee of obtaining the best solution, in accordance with the principles inherent in stochastic techniques. In contrast to the EBQO technique, the effectiveness of GBQO may not always result in the highest quality solutions [24]. The inherent trade-offs present in the field of optimisation approaches are highlighted by this acknowledgement. A pseudo-code version of the proposed SBQO operational core has been meticulously developed to capture the nuances of the methodology as presented in Algorithm 1. This pseudo-code acts as a foundation for direction, illustrating the anticipated series of actions essential to the SBQO optimisation procedure. The pseudo-code also appears as a light, beckoning additional investigation and improvement within the field of query optimisation tactics in distributed systems, beyond its procedural explanation.

Algorithm 1: Proposed SBQO Algorithm for Query Processing

```
1.  Input: Read DSS query, no. of base relations, fragments, no. of operations, I/O costs, Processing cost
2.  Output: Total cost, run-time, reduction rate of processing cost
3.  Initialize:
4.  $Randomly_{Generate} \leftarrow$ Initial Population
5.  $Design_{Chromosome} \leftarrow Length = 1 < (No.of\ operations)$
6.  Compute:
7.  $H(f) = \frac{1}{1-\theta}\sum_{k=1}^{n} f^k - 1$
8.  Perform crossover and mutation operation
9.  Assess fitness:
10. $T(Cost_{DSS}) = T(Cost_{I/O}) + T(Cost_{CPU}) + T(Cost_{comm})$
11. Continue till Maximum no. of generations
12. Exit
```

6. Experimentation and Results Analysis

A carefully selected set of ad hoc queries has been developed in order to evaluate the effectiveness and performance of several query optimizers within a Distributed Decision Support System (DSS) framework. The TPCDS benchmark database, which is closely entwined with customer- and sales-related statistics, is the focus of these precisely crafted queries [30]. These questions are painstakingly expressed as relational algebraic expressions.

The collection of queries has been carefully designed to include a range of join operations ranging from one to ten, providing an extensive spectrum for experimental study. The number of join operations can be changed for analytical diversification thanks to the clever construction of this query ensemble. The query set efficiently interacts with a collection of relational entities including Customer, Sales, Cust_Address, Marketing, Shipping, Webstore, Warehouse, Store, and Items when run inside the framework of a distributed database environment [30].

A sophisticated simulator was painstakingly created to unravel the intricacies involved in this work within the context of distributed DSS inquiries in order to meet the complex challenge of operation site allocation. This simulator was cleverly designed using the MATLAB 2008 environment, painstakingly constructed without using the built-in "GA" (Genetic Algorithm) features, and thus encapsulates a uniquely customised approach. The population size for the GA has considered at size 50, number of generations at 50, crossover probability of 0.3, along with mutation probability of 0.02, respectively. This system's main goal is to ingest the complex parameters of a DSS query and then produce a wide variety of query execution strategies as an output.

This simulator's functionality depends on a well chosen set of input parameters that have all been properly calibrated to coordinate the optimisation process for a DSS query. These include crucial elements like the number of base relations, the total number of operations, the number of projection and selection operations, the number and size of intermediate fragments, and the coefficients for estimating the cost of I/O, communication, and processing. Notably, the quantity of join operations also contributes significantly to this optimisation process. The simulator works to identify the best query allocation strategy through a complex optimisation procedure, effectively minimising the aggregate use of important computational resources like I/O, CPU processing, and communication. The system's ultimate result is this carefully chosen allocation design, which has been refined through painstaking optimisation. It is crucial to remember that all experimental iterations were conducted under a clear set of assumptions, as explained in other academic studies [13,14].

The calculations carried out for this study were based on the careful identification of data block requirements for certain queries. It's noteworthy that an 8 KB standard block size was proposed for each relation's dimensions. Relevantly, the dispersed aspect of the design was strengthened by the foundational base relation being randomly repeated across two different sites. A crucial aspect, the size of intermediate pieces, was carefully determined through the use of selectivity estimation approaches. The default proportionality between cost coefficients for input-output and communication efforts was set at the canonical ratio of 1:1.6 in accordance with prevalent norms. Effective "Selection" and "Projection" activities were deftly agreed to only take place at the locations housing the relevant basis relations, ensuring the best locality-aware execution. In stark contrast, the strategic execution of 'Join' operations was given the flexibility to unfold at any point over the vast expanse of the distributed database network.

A carefully crafted collection of dispersed queries was conceptualised for this inquiry, built on the foundation of ad-hoc DSS inquiries. The core of these inquiries' themes is found in the realm of retrieval operations, more specifically in the complex environment of a distributed database system. These questions highlight the importance of the "Join" operation in the context of distributed database queries by combining it with selection, projection, and join procedures from the field of relational algebra. They also illustrate the philosophy of ad hoc analysis. The query collection was meticulously organised across several levels of join complexity to fully represent the gamut of join procedures. The investigation's next phase consisted of a series of meticulous tests carried out on this collection of dispersed DSS queries.

The 'Costs Model' put forth by eminent researchers like Rho and March, Dougless and Cornell, Sevinc and Cosar [8,14,28] serves as the foundation upon which the costing dynamics are constructed. In terms of costing coefficients, the design manifests as a linear array of "Input-Output" costs, with the number of sites integrated in the distributed database architecture limiting its dimension. The idea of cost equivalence is upheld, with the ratio of 'Input-Output' costs coefficients to communication costs coefficients skillfully set at 1:1.6. This is done by leaning on the authoritative contributions of Rho and March, Sevinc and Cosar, and Ozsu and Valduries. The communication costs coefficients are noteworthy; they are painstakingly expressed as a square matrix, with the size of the matrix being methodically controlled by the number of sites interacting with the architecture.

Additionally, a rigorous calibration of a 1:10 proportionality nexus between processing cost coefficients and input-output cost coefficients is made. The architecture of processing costs coefficients manifests as a linear array, akin to 'Input-Output' costs coefficients. It's important to emphasise that this composite array provides a prototype that includes various costs coefficients relevant to a DSS query and is harmoniously positioned within a distributed database system made up of ten different sites.

A systematic framework is established to ascertain the localised processing costs (referred to as $Cost_{local}$) and communication costs (referred to as $Cost_{comm}$) associated with a given DSS query by using the meticulously created decision variables and the intricate cost coefficients described earlier. To clarify further, $T(Cost_{I/O})$ is for total input-output costs, whereas $T(Cost_{CPU})$ stands for total query processing costs.

The sum of all input-output costs $T(Cost_{I/O})$ and all processing costs $T(Cost_{CPU})$ related to the collection of selection, projection, and join operations that make up the query are combined to determine the localised processing costs. In more specific terms, the input-output cost coefficients (abbreviated as $Cost_{I/O}$) associated with a given site are multiplied by the number of memory blocks accessed via a given base relation b, effectively reflecting the size of intermediate fragments, to determine the input-output costs attributable to the selection operation. Parallel to this, the number of memory blocks that are either read from or written to by a certain base relation b is multiplied by the processing cost coefficients (termed as $Cost\_Coeff_{CPU}$) associated to the chosen site to obtain the "Processing Costs." In conclusion, summative computations coordinate the fusion of individual contributions, which ultimately results in the determination of both the total input-output costs and the overall processing costs related to the current query.

The area of join operations is where the dimension of communication expenses is most relevant. Through a series of carefully considered stages, as listed below, the quantification of communication costs related to a specific inquiry is painstakingly determined:

1.) The communication costs from the relevant site hosting the left child of the "Join" operation are detected in the initial step of computation. The number of data blocks specified by the 'Left Fragment' component of the specific join operation is multiplied by these communication expenses after that.
2.) The right child of the 'Join' operation is parallel-calculated in a manner similar to the step before. In this case, the communication costs associated with the location of the join operation are accurately determined and then scaled by the number of data blocks assigned to the relevant join operation's right fragment segment.
3.) The concluding phase entails a cumulative integration of the results obtained from the computations above. This summative procedure develops repeatedly in accordance with the total number of join operations woven into the current query's structure.

These calculations carefully explain the complex physics underlying the estimation of communication costs in the context of join operations, encompassing the dynamic interactions between various query structure parts.

A precise mathematical formulation that clearly separates the calculation of local processing costs and communication costs serves as the conceptual foundation for the proposed cost model. This phrase is carefully and precisely expanded upon in the next explication.

$$Cost_{local} = \sum_i \left(Cost_{I/O} \times \alpha_i\right) + \sum_j \left(Cost_{I/O} \times \alpha_j\right) + \sum_i \left(Cost\_Coeff_{CPU} \times \alpha_i\right) + \sum_j \left(Cost\_Coeff_{CPU} \times \alpha_j\right)$$
(1)

An exacting mathematical representation that is robust and precise is used to convey the complex quantification of communication costs (CMCT). The explanation that follows painstakingly expands on this mathematical formulation reveals the complex complexities that underlie the computation of communication costs within the boundaries of the specified research area.

$$Cost_{comm} = \sum_i Cost\_Coeff_{comm}\left(L_{prev\_join}, loc_{join\_op}\right) \times L_{prev\_frag} + \sum_i Cost\_Coeff_{comm}\left(L_{prev\_join}, loc_{join\_op}\right) \times R_{prev\_frag} \quad (2)$$

Where,

- $Cost\_Coeff_{comm}$= cost coefficient for communication
- L_{prev_join}= Left previous operation
- loc_{join_op}= location of the join operation
- L_{prev_frag}= left previous fragment
- R_{prev_frag}= right previous fragment

Thus, the total cost for the DSS can be obtained through the formulation below:

$$T(Cost_{DSS}) = \sum_i\left(Cost_{I/O} \times \alpha_i\right) + \sum_j\left(Cost_{I/O} \times \alpha_j\right) + \sum_i\left(Cost\_Coeff_{CPU} \times \alpha_i\right) + \sum_j\left(Cost\_Coeff_{CPU} \times \alpha_j\right) + \sum_i Cost\_Coeff_{comm}\left(L_{prev\_join}, loc_{join\_op}\right) \times L_{prev\_frag} + \sum_i Cost\_Coeff_{comm}\left(L_{prev\_join}, loc_{join\_op}\right) \times R_{prev\_frag}$$
(3)

The approaches covered—GBQO, EBQO, and proposed SBQO—are each put through a variety of carefully planned experiments where the systematic alteration of genetic approach parameters is carried out. Variable parameters include things like population size, the number of generations, crossover rates, and mutation rates as discussed above. An optimal configuration, defined by the most advantageous "Total Costs" values, is discovered by the orchestration of genetic parameters, according to empirical research. This crucial realisation offers a solid basis for identifying the genetic parameter statistics that are thought to be most effective for achieving the desired results within the parameters of the aforementioned distributed database queries [20,21].

6.1. Result Analysis

The stochastic DSS query optimizer is analysed and improved as part of this research, which follows the methodology stated in the reference [32]. A set of methodical experiments are meticulously carried out with the goal of optimising a selected group of distributed DSS queries. The overriding goal of these efforts is to reduce the demand on system resources, enabling the efficient execution of the optimised queries.

The use of system resources in the context of a distributed DSS query falls into three categories: input-output, processing, and communication. The manifestation of "Total Costs," which is frequently used interchangeably with "Total Time," is the culmination of this comprehensive amalgamation of resource utilisation. This total indicator captures the overall use of the system resources required for the query's execution [13]. The enhancement of throughput within the stochastic query optimizer is the main emphasis of this study.

The following essential aspects are in line with this crucial goal:

1. *Analysis of Several Meta-heuristic-based Query Optimizer:* The EBQO, GBQO, and SBQO are just a few of the DSS query optimizers that will be carefully dissected and evaluated as part of this research. This analytical framework tries to break down the advantages and disadvantages of each optimizer, providing a thorough understanding of their effectiveness in relation to query optimisation.
2. *Impact of Data Replication Factor:* The thorough assessment of the impact of the data replication factor on the complex procedure of DSS query optimisation is an important aspect of this investigation. This investigation dives into how different amounts of data replication affect optimisation dynamics, providing essential insights into how replication and optimisation results interact.
3. *Statistical Analysis*: The paper conducts a detailed statistical investigation of the complex interaction between the count of join operations and the ensuing use of system resources required for the execution of distributed DSS queries. This investigation aims to identify the observable dependencies and patterns that support this important component of query optimisation.

This research approach emphasises a multifaceted project that includes analysis, augmentation, and in-depth inquiry and is centred in the area of distributed systems query optimisation.

Figure 2 is a diagram that illustrates the various sets of "Total Costs" for a selected group of distributed DSS queries. The GBQO, EBQO, and proposed SBQO are the three different approaches used to optimise queries. Notably, the benchmark outcomes obtained using the EBQO and GBQO approach are contrasted with the proposed alternative. Beyond the numerical representation, Figure 2 goes further to provide an assessment of the solution quality in relation to the "Total Costs," placing the outcomes in the context of the exhaustive enumeration paradigm. Along with this, Figure 2 supports the claim by emphasising the key findings from the investigation.

The statistics show a clear pattern: when compared to the EBQO, solutions to the "Operation Site Allocation Problem" in DSS queries that were realised using the SBQO approach display a relative sub-optimality of about 20%. The GBQO put forth by Sevinc and Cosar represents a significant advancement because it raises the quality of solutions as determined by "Total Costs" by up to 5% inside the framework of SBQO. The results of the GBQO, which is the subject of this analysis, reveal a 15% difference in optimality when compared to the standard set by the EBQO. The outcomes of the NBQO are incrementally improved by about 3% as a result of the GBQO methodology's optimisation efforts.

Interestingly, despite the release of the SBQO, which steadily improves the quality of solutions obtained through 5%, the trajectory of refinement continues. As a result, the results produced by SBQO are very similar to those realised by EBQO. Therefore, the inclusion of the SBQO results in a startling alignment of solution quality, as measured by 'Total Costs,' with the benchmark standard set by the EBQO technique. Effectively reducing the gap, this optimisation methodology produces results that are comparatively more high-quality than those obtained by the EBQO.

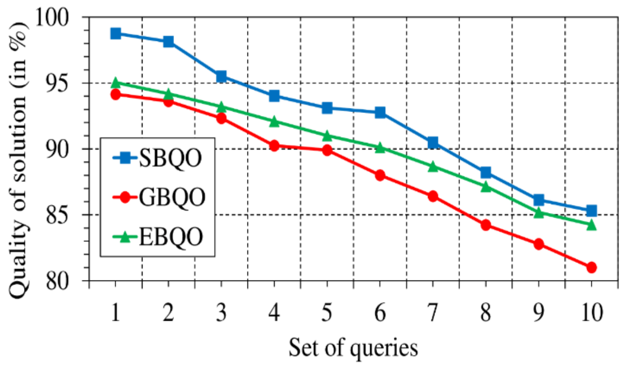

Fig 2. Comparison of quality of solution in % for the proposed SBQO algorithm with GBQO and EBQO algorithms for different set of queries.

The runtime requirements necessary to provide an ideal solution for the operation site allocation problem inherent to DSS questions are illustrated in Figure 3 as an example. The temporal demands imposed by various optimisation strategies are methodically contrasted and contextualised in this graphic representation. The conclusions drawn from Figure 3 come together to form a unified illustration: the EBQO technique is revealed to be a good fit for simple DSS queries. This results from the finding that when faced with more complex and large-scale DSS queries, the runtime for EBQO demonstrates an exponential increase. The runtime trajectories linked to GBQO and SBQO, on the other hand, reveal a distinctive pattern, either maintaining a state of essentially constant behaviour or demonstrating a noticeably progressive rise. This empirical trend highlights the scalability and robustness of these stochastic evolutionary techniques, making them effective tools regardless of the complexity of the query or the number of join operations.

Figure 3 also reveals an unexpected finding: all stochastic query optimizers' temporal profiles converge onto a single trajectory. This convergence is a sign of a stable behaviour where there are barely perceptible variations in their individual runtime values. This uniformity emphasises how stable and dependable these stochastic optimisation techniques are. The primary takeaway from these findings is that, while the EBQO approach is still applicable for basic DSS queries, its viability decreases significantly as queries become more sophisticated. Stochastic techniques, as represented by GBQO and proposed SBQO, stand out as strong competitors, continuously providing respectable runtimes regardless of the complexity of the query or the number of join operations involved. This empirical investigation supports the idea that stochastic query optimizers provide a flexible and reliable answer to a variety of questions.

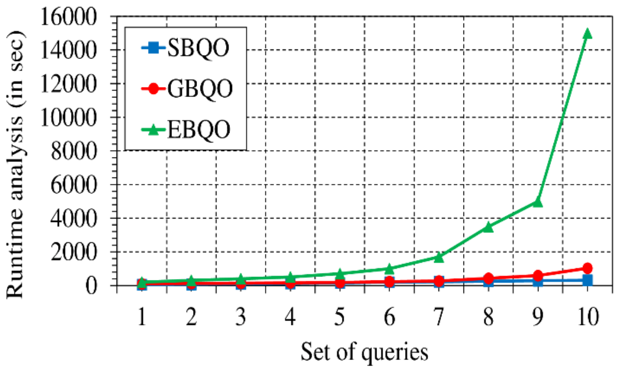

Fig 3. Comparison of runtime in seconds for proposed SBQO algorithm along with GBQO and EBQO algorithms for different set of queries.

The precise goal of a series of rigorously carried out experiments was to investigate the impact of data replication rates on the quantity of system resources required for the smooth execution of distributed DSS queries. The measurement of the 'Total Costs' associated to these distributed DSS queries, which acts as a complete indicator that includes resource use, is crucial to this endeavour. The EBQO, GBQO, and the SBQO algorithms have all been used in these research.

These experimental iterations' empirical findings have led to some noteworthy revelations. As data replication rates increase from 20% to 90%, a recognisable pattern starts to show. Within the optimisation of the 'Total Costs' of distributed DSS queries, a discernible improvement appears in this trajectory. Through the use of GBQO and SBQO optimisation approaches, the increase in replication rate specifically catalysed reductions in "Total Costs" of 2%, and

3.5% over a selected set of experimental DSS queries. Figure 4, a visual explanation that depicts the influence of increased replication factors on various stochastic DSS query optimizers, provides a vivid illustration of this empirical trajectory. The observed pattern supports the claim that a SBQO produces the best results when queries are optimised with a reliable 90% replication factor. Therefore, these empirical findings highlight the incremental optimisation potential resulting from increased data replication rates and support the improved efficiency of various stochastic DSS query optimizers. Additionally, this empirical investigation highlights the SBQO's superior performance when faced with large replication rates, demonstrating its effectiveness in resource-constrained circumstances.

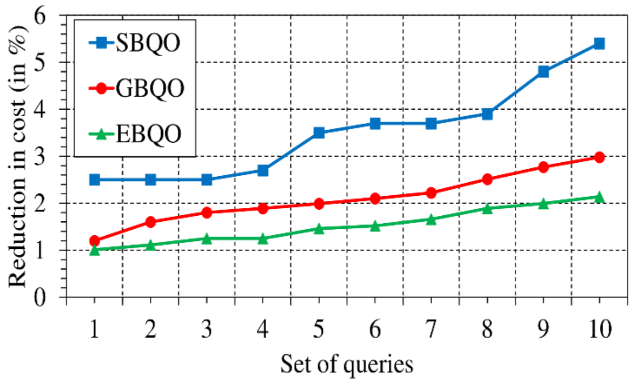

Fig 4. Comparison of reduction in cost incurred in % for proposed SBQO algorithm with GBQO and EBQO algorithms for different set queries.

7. Conclusions and Future Work

The handling of substantial data volumes, frequently spanning GigaBytes, PetaBytes, or even beyond, presents a significant challenge in the domain of DSS queries. As a result, the optimisation of DSS queries revolves around a fundamental metric known as "Total Costs," a composite amalgamation of various costs associated with query processing in large databases. The main goal of this study is to improve the performance of distributed DSS queries, which leads to the creation of a model that is integrated into the SBQO algorithm. This innovative approach aims to speed up the creation of efficient allocation schemes for query operations, ultimately speeding up the overall optimisation process. To address the challenging "operation site allocation problem" that arises with distributed DSS queries, a specialised simulator is developed. This effort is supported by a comprehensive empirical assessment, which employs a variety of query optimisation approaches, such as GBQO and EBQO, alongside the proposed SBQO algorithm to carefully examine a number of ad-hoc DSS queries in terms of their total costs, runtime outcomes, and reduction in cost factor.

A careful investigation reveals some interesting findings. It is noteworthy that when compared to the SBQO strategy, both GBQO and EBQO display poor performance in achieving optimal query execution plans. The development of the SBQO approach, which substantially improves the solution quality of GBQO and EBQO by margins of 4.61% and 3.72%, respectively, effectively fills this gap. Additionally, a novel twist is provided, including the idea of entropy proposed by Havrda and Charvat into the stochastic framework, to increase the effectiveness of stochastic query optimizers. Empirical findings highlight the value of this addition, which is supported by the improved performance of the SBQO technique.

A significant element is the complex interaction between the distributed database system's intrinsic replication feature and the resulting "Total Costs" of DSS queries. Notably, increasing the replication rate from 20% to 90% significantly lowers "Total Costs" across several techniques, respectively. A thorough statistical analysis further highlights the strong association between the number of join operations and the 'Total Costs' of DSS queries.

The implementation of targeted research initiatives aiming at automating the transformation process is necessary for further development in this field. This procedure involves creating a query tree from a "SQL"-based query without error as a prelude to starting the optimisation process. By shortening the initial phases, this automation would significantly improve the optimisation pipeline's effectiveness and seamlessness. A careful analysis of various selection strategies within the genetic approach framework can be used to fine-tune the evolutionary progression of the entropy-based stochastic DSS query optimizer. By dissecting the sub tleties and dynamics of various selection strategies, this analytical examination hopes to improve the suggested model's ability to optimise. An evaluation of the suggested approach in comparison to previous nature-inspired evolutionary optimisation strategies is necessary to show its uniqueness and effectiveness. A thorough analysis of the advantages and distinctive features of the entropy-based stochastic DSS query optimizer can be carried out by placing the results in the larger framework of evolutionary optimisation techniques. Furthermore, a thorough investigation of the complex interplay between data allocation and access policies is necessary. For DSS query optimisation, this dimension has significant ramifications. In order to fully understand the complex repercussions of various data allocation schemes and access policies, as well as their resulting impact on optimisation outcomes, a systematic analysis is essential.

Acknowledgements

The authors are thankful to the anonymous reviewers for their valuable suggestions, which greatly assisted in

improving the quality of the manuscript.

Author contributions

Sambit Kumar Mishra: Conceptualization, Methodology, Software, Field study **Anita Mohanty:** Data curation, Writing-Original draft preparation, Software, Validation., Field study, Visualization, Investigation, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest with the publication of this article.

Reference

[1] Hevener AR, Yao SB. Query processing in distributed database systems. IEEE Trans. Softw. Eng. 1979;5(3):177–87.

[2] Ceri S, Pelagatti G. Allocation of operations in distributed database access. IEEE Trans. Comp. 1982;31(2):119–29.

[3] Chen Yan, Zhou Lin, Li Taoying, Yu Yinging. The semi-join query optimization in distributed database system. In: National Conference on Information Technology and Computer Science. Atlantis Press; 2012. p. 606–9.

[4] Martin TP, Lam KH, Russel Judy I. An evaluation of site selection algorithm for distributed query processing. Comp. J. 1990;33(1):61–70.

[5] Apers Peter MG, Hevner Alan N, Yao Bing S. Optimization algorithms for distributed queries. IEEE Trans. Softw. Eng. 1983; SE-9.1:57–68.

[6] Ghaemi Reza, Fard Amin Milani, Tabatabaee Hamid, Sadeghizadeh Mahdi. Evolutionary query optimization for heterogeneous distributed database systems. World Acad. Sci., Eng. Technol.2008;2:34–40.

[7] Mor Jyoti, Kashyap Indu, Rathy RK. Analysis of query optimization techniques in databases. Int. J. Comp. Appl. 2012;47(15):5–9.

[8] Sevinc Ender, Cosar Ahmat. An evolutionary genetic algorithm for optimization of distributed database queries. Comp. J. 2011; 54:717–25.

[9] Kayvan Asghari, Ali Safari Mamaghani, Mohammad Reza Meybodi, An evolutionary algorithm for query optimization in database, in: Innovative Techniques in Instruction, E-Learning, E-Assessment and Education, 2008, pp. 249–254.

[10] Chande Swati V, Sinha Madhvi. Genetic algorithm: a versatile optimization tool. BVICAM's Int. J. Inf. Technol. 2008;1(1):7–12.

[11] Panicker Shina, Vijay Kumar TV. Distributed query plan generation using multi-objective genetic algorithms. World Scient. J.2014; 2014:1–17.

[12] Johann Christoph Fregtag, The Basic Principles of Query Optimization in Relational Database Management System, European Computer Industry Research Centre Germany, Internal Report IR-KB-59, 1989, pp. 1–15.

[13] M. Tamer Ozsu, Valduries Patrick, Principles of Distributed Database System, second ed., Pearson Education (chap. 1–6).

[14] March, Rho ST. Allocating data and operations to nodes in distributed database design. IEEE Trans. Knowl. Data Eng. 1995; 7(2):305–17.

[15] Kumar TV, Singh V, Verma AK. Distributed query processing plan generation using genetic algorithm. Int. J. Comp. Theory Eng. 2011;3(1):38–45.

[16] Goldberg David E. Genetic Algorithm in Search, Optimization & Learning. New Delhi: Pearson Education; 1999 (chap. 1).

[17] Paulinas Mantas, Usˇinskas Andrius. A survey of genetic algorithms applications for image enhancement and segmentation. Inf. Technol. Control 2007;36(3):278–84.

[18] Carlos Alberto Gonzalez Pico, Roger L. Wainwright, Dynamic scheduling of computer tasks using genetic algorithms, in: Proceedings of the First IEEE Conference on Evolutionary Computation IEEE World Congress on Computational Intelligence, Orlando, 1994, pp. 829–833.

[19] Omara Fatma A, Arafa Mona M. Genetic algorithm for task scheduling problem. J. Paral. Distrib. Comput. 2010;70(1):13–22.

[20] Karegowda Asha Gowda, Manjunath AS, Jayaram MA. Application of genetic algorithm optimized neural network connection weights for medical diagnosis of Pima Indians diabetes. Int. J. Soft Comput. 2011;2(2):15–23.

[21] Hill Anthony M, Kang Sung-Mo. Genetic algorithm based design optimization of CMOS VLSI circuits. Lecture Notes in Computer Science 2005;866:545–55.

[22] Lienig J. A parallel genetic algorithm for performance-driven VLSI routing. IEEE Trans. Evolution. Comput. 1997;1(1):29–39.

[23] Man KF, Tang KS, Kwong S. Genetic algorithms: concept and applications. IEEE Trans. Indust. Electron. 1996;43(5):519–34.

[24] Du Jun, Alhajj Reda, Barker Ken. Genetic Algorithm based approach to database vertical partition. J. Intell. Inf. Syst. 2006;26:167–83.

[25] Kapoor JN. Measures of Information and Their Applications. Wiley Publishers; 1994.

[26] Zhou Rongxi, Cai Ru, Tong Guanqun. Applications of entropy in finance: a review. Entropy 2013;15(11):4909–31.

[27] Hien To, Kuorong Chiang, Cyrus Shahabi, Entropy-based histogram for selectivity estimation, in: CIKM, 2013, pp. 19391948.

[28] Cornell Douglas W, Yu Philip S. On optimal site assignment for relations in the distributed database environment. IEEE Trans. Softw. Eng. 1989;15(8):1004–9.

[29] Pramanik Sakti, Vineyard David. Optimizing join queries in distributed databases. IEEE Trans. Softw. Eng. 1988;14(9): 1319–26.

[30] TPS-DS Benchmark Report, 2012 <www.tpc.org/tpcds/spec/tpcds_1.1.0.pdf> (accessed on 25/04/2013).

[31] Sarjo, Kapila, Kumar Dinesh, Kanika. A genetic algorithm with entropy based probabilistic initialization and memory for automated rule mining. Adv. Comp. Sci. Inf. Technol. Commun. Comp. Inf. Sci. 2011;131:604–13.

[32] Drenick PE, Smith EJ. Stochastic query optimization in distributed databases. ACM Trans. Database Syst. 1993;18(2): 262–88.