International Journal for Housing Science and Its Applications
Publish August 10, 2025. Volume 46, Issue 4 Pages 4188-4202

https://doi.org/10.70517/ijhsa464271

Research on Efficient Database Query Optimization Algorithm Based on Distributed Computing Model

Xingyan Shi[1,*]
[1] Faculty of Information Engineering, Henan Vocational College of Agriculture, Zhengzhou, Henan, 451450, China
Corresponding authors: (e-mail: 13849029903@163.com).

Abstract Existing distributed database multi-connection optimization searches have high space complexity, which greatly affects the search efficiency. In this paper, we propose a parallel genetic-maximum minimum ant colony algorithm of (PGA-MMAS) for query optimization. Firstly, based on the faster convergence of genetic algorithm (GA), initialization coding and optimal population screening operations are performed on the final connectivity relations to obtain a set of better query execution plan (QEP), and the better QEP is transformed into the initialized pheromone distribution of maximal minimal ant colony algorithm, and the updating and cycling of the pheromone matrix is performed according to pheromone updating rules, and the global optimal QEP is finally searched more rapidly. The improved crossover operation of this paper's optimization algorithm improves the solution efficiency by 10%, and the total execution time is also greatly reduced compared with the three compared algorithms. The query latency of this paper's algorithm is reduced compared to the SparkSQL algorithm on five different experiments, and the reduction is 60% and above. It shows that the algorithm in this paper can improve the query efficiency of distributed databases and ensure a more efficient query method.

Index Terms Genetic Algorithm, Maximum Minimum Ant Colony Algorithm, Distributed Database, Query Optimization

I. Introduction

In recent years, with the development of computer networks and database technology, the application of distributed databases has become more and more widespread. As the application continues to expand, the query of data is also more and more complex, the efficiency of the query requirements are also more and more high, so the query processing has become a key issue in the distributed database system [1]-[4]. In distributed databases, due to the distribution and redundancy of data, query processing generally requires inter-site data transfer and communication costs, which becomes a major contradiction in query optimization [5]-[8]. On the other hand, the distribution and redundancy of data also increase the possibility of concurrent processing of queries, which can shorten the response time of query processing and improve the processing speed [9]-[11].

Query optimization in distributed database systems has two different objectives, one objective is to minimize the total cost and the other objective is to minimize the query response time, which is of great importance in distributed database systems [12]-[15]. Because a distributed database system is a system composed of multiple computers, the distribution and redundancy of data also increase the possibility of parallel processing of queries, which can shrink the response time of query processing and speed up the query processing [16]-[19]. It is also common to use both criteria in distributed query optimization, one as the primary criterion and the other as a secondary criterion depending on the system application [20], [21]. In distributed database systems, query optimization consists of two elements, including query strategy optimization and local processing optimization, while query strategy optimization is particularly important, the advantages and disadvantages of distributed query strategy will directly affect the amount of computer network resource consumption [22]-[25]. Currently, the basic methods for optimizing distributed database queries include optimization algorithms based on semi-connected operations, optimization algorithms based on direct-connected operations, optimization algorithms based on query graphs, and genetic algorithms [26], [27].

Literature [28] pointed out that the query optimization problem in large-scale distributed databases is the NP-hard problem and that this problem is difficult to solve, and reviewed various optimization strategies to show that the ant colony optimization algorithm, in combination with other optimization algorithms, can effectively improve the performance of distributed query optimization. Literature [29] develops a distributed database index query optimization method based on consistent hash algorithm, which aims to solve the problem of load imbalance in distributed database index query, and shows that the method can effectively adjust the load distribution of distributed

database with strong query response capability. Literature [30] examined the design of a cost-based query optimizer and concluded that a hybrid approach is an effective way to cope with the query optimization problem, and that it is capable of accomplishing a large number of query optimization tasks through the use of genetic algorithms and swarm particle optimization methods. Literature [31] emphasized that query optimization is the most important part of distributed database models and used quantum-inspired ant colony algorithms to improve the cost of query joins in a distributed database model, revealing that the model performs better with the same number of ants and has good convergence performance. Literature [32] and literature [33] proposed an artificial bee colony algorithm for the query optimization problem in distributed database systems, revealing that the algorithm achieves lower execution cost and faster convergence speed, which can effectively improve query efficiency, but requires more execution time. Literature [34] examines distributed databases in cloud environments and highlights the important role of query optimization mechanisms to improve performance in cloud environments and reviews different techniques for query optimization in order to provide researchers with a complete view of query processing and its optimization techniques. Literature [35] constructs a mathematical model for query optimization of distributed database and proposes an adaptive genetic algorithm based on double entropy and verifies the effectiveness of the algorithm in query optimization of distributed database. Literature [36] attempted to design a quantum-inspired ant colony based algorithm to improve the cost of query joins in distributed databases, revealing that the algorithm has a faster convergence speed and better convergence results compared to the classical model. Literature [37] describes the Multi-objective Genetic Algorithm with Bat, which is a hybrid of Multi-objective Genetic Algorithm and Bat Algorithm, which improves the quality of query plan by using it in generating the optimal query plan, but it takes longer time. The above study examined the algorithms such as Ant Colony Optimization Algorithm, Hash Algorithm, Artificial Bee Colony Algorithm for optimizing the query optimization of distributed databases, which are effectively applied to improve the convergence speed and reduce the cost.

In this paper, we first apply a strategy selection algorithm to reduce the complexity of the search space, and propose a database query optimization algorithm that combines the genetic algorithm and the max-min ant colony algorithm. The genetic algorithm is utilized to transform the tree structure in the relationally connected sequential coding segment to obtain a structured coding. To ensure the smooth fusion of the two algorithms, an adaptive dynamic fusion method is adopted to bridge the two algorithms at the optimal moment. The genetic algorithm outputs the globally better solution of the whole population to the max-min ant colony algorithm, which then introduces the global update rule to search for the optimal solution until the globally optimal QEP is obtained. Finally, performance test experiments and example experiments are designed to verify the solution performance and application effect of the optimization algorithm in this paper.

II. Database queries based on a distributed computing model

II. A. Computational Model of Query Engine for Distributed Database Systems

Traditional relational databases, including Oracle, MySQL, and SQLServer, provide users with powerful yet simple data manipulation semantics - SQL - and still play an important role in the market today. However, with the advent of the big data era, traditional stand-alone databases have become increasingly unable to meet the market demand in terms of computational and storage capacity, thus giving rise to a variety of distributed data processing platforms.

In the traditional database a SQL statement needs to go through the following process, SQL statement first through the lexical parsing syntax parsing to generate a syntax tree, the syntax tree is actually used to describe the logical process of the execution of the SQL statement. After the SOL query optimizer, the syntax tree is deformed to get the deformed syntax tree. The transformed syntax tree is handed over to the database execution engine, which generates the corresponding execution plan, and finally the execution engine executes the plan and returns the result to the user.

For most databases, the process from parsing SOL statements to generating optimized SOL syntax trees is similar, the difference is that different database query engines generate very different execution tasks.

II. A. 1) Generalized computational models

In processing and analyzing large-scale data sets, Hadoop has achieved great success with Hdfs for unstructured data storage, HBase for structured data storage, Map/Reuce, Tez for computational processing, and Yarm or Mesos for resource management platforms. In terms of big data processing, Hadoop has constituted a good and sustainable ecosystem, so a good way to extend the database system is to extend the functionality of the database system based on the Hadoop platform.

Hive is the first system that appeared in Hadoop platform and provides SQL-like operations, undoubtedly, its query engine adopts Map/Reduce, Hive parses the SQL statement into a syntax tree, optimized according to certain optimization rules, and then converted into the corresponding Map/Reduce execution tasks.

In later development, Hive adopted the relatively high performance Tez as the query engine, and Shark itself abandoned the Hive parser and rewrote the SQL parsing and optimization module to produce the current Spark-SQL. These distributed database systems, which are built on top of a general-purpose computing engine, make full use of the advantages of the current open source big data processing platforms, and greatly expand the database system on big data. These distributed database systems are built on top of general-purpose computing engines, taking full advantage of the current open-source big data processing platforms and greatly expanding the processing capabilities of database systems on big data.

II. A. 2) Specialized computational models

In addition to relying on the Hadoop ecosystem to build distributed databases, the database query engine itself is also exploring and developing in the direction of distributed and parallel computing. Early traditional database systems based on the volcano model, that is, the syntax tree is converted into physical operators, each physical operator to perform specific operations, line by line, line by line processing data, and line by line, line by line, line by line, the processed data will be output to the subsequent operator until the entire computation process is complete. Traditional databases such as Mysql, Oracle, SQLServer, etc. are based on such a set of physical operator rules to build the query engine. Since this type of query engine in the process of processing to the data line unit for processing, we can attribute it to the rows of query engine [38].

II. B. Multi-connected Query Optimization in Distributed Databases

A Multi-Join Query (MJQ) [39], is an expression containing multiple relations formed after a series of projection and selection operations. The multi-connection query optimization in the traditional database often only involves the number of relationships less than 10, and the number of tuples contained in each relational table is also acceptable and coping with in the traditional database processing, but with the popularization of database applications and the rapid development of the Internet, the huge amount of data that has increased rapidly makes the current single query also contain the number of relationships and connections that the traditional query optimization technology is not enough to handle, which greatly increases the complexity of multi-connection query optimization and the difficulty of solving this problem.

II. B. 1) Representation of Multi-connected Queries in Distributed Databases

Throughout the multi-connected query optimization process, the query optimizer first uses the nodes and edges in the query graph G = (V, E) to represent the relations and connections between relations involved in a join, and then transforms the query graph into many different join trees representing all equivalent expressions. In this paper, we define connected query graphs to better characterize the representation of multi-connected queries in distributed databases.

Definition 1: Connected graph.

Given a query Q, its join graph is defined as $G_Q = (V, E)$, where, if table T_i is in Q, there is a node n_i representing this table in V.

If the equivalent join of T_i and T_j is a join operation in Q, an undirected edge $e = (n_i, n_j)$ is created and the label of the edge e is set to k.

When considering only the case of natural joins, it is straightforward to use the names of the same attributes of two relations to indicate that a connection exists between the two relations.

The end result of the optimization of multi-connection query logic for distributed databases is the least costly order of execution of connections represented by some join tree.

II. B. 2) Search Space for Multi-Join Queries in Distributed Databases

In this paper, we introduce logical and physical search spaces for multi-connection query optimization for distributed databases, respectively, to facilitate the discussion on query optimization search spaces. The commonly used logical search space is discussed first.

Before discussing the logical search space, this paper first gives a brief introduction to the concept of join tree.

Join trees are categorized as left linear trees, right linear trees, and dense trees. In this paper, we have chosen to use LS, RS, and BS as the abbreviations for each of the three linear trees, and AS as the abbreviation for the entire search space.

For the logical optimization of a multi-connected query for a distributed database containing N connection operations, all its equivalent algebraic expressions (connection trees) represent the entire policy space in its entirety. In the optimization of a query containing N joins, a binary join tree representing a certain execution plan is composed using the number of N join operations as the number of parent nodes and the number of N+1 relations involved in the join as the number of leaf nodes. N different nodes can be arranged and combined to form $C_{2N}^{N} / N+1$ binary

tree with different morphology, therefore, the size of the strategy space for logical optimization obtained by the binary tree containing N+1 leaf nodes after deformation of all possible scenarios is:

$$\frac{1}{N+1}C_{2N}^{N} * (N+1)! = C_{2N}^{N} * N! \tag{1}$$

Therefore, it is derived that the sizes of LS, RS, and AS are $(N+1)!,(N+1)!,C_{2N}^{N} * N!$. In addition, the BS is calculated to be AS-LS-RS.

In distributed databases, the variety of physical operations supported by the query execution engine and the execution efficiency determine the size of the strategy space and the optimization effect of the physical optimization of multi-connected queries, respectively. The physical algorithms that support natural join execution are generally four kinds: nested loops, indexing or hashing, subsumption sorting, and hashing.

II. B. 3) Strategy Selection

Even if the number of relations involved in joins is small, the size of the search space for optimizing multi-connected queries for distributed databases is a large and alarming number. Therefore, to further optimize multi-connection queries, we must first start by simplifying its search space, reducing the number of home join trees in the search space, and reducing the complexity of the search space.

Definition 2: Coverage set of a join graph.

Given a connected graph $G_Q = (V, E)$, its connected set is a series of graphs which satisfy:

$\forall G_i \in S$, G_i is a subgraph of G_Q. That is, given a node n_x of G_i and an edge e_y, $n_x \in V, e_y \in E$.

$\forall G_i \in S$, if n_x and n_y are two nodes of G_i, there must be an edge in G_i that connects n_x and n_y.

$$G_Q V = \bigcup_{vG_i \in S} G_i V \tag{2}$$

$\forall G_i, G_j \in S \to G_i, V \bigcap G_j Y = \varnothing$. i.e., there are no common nodes in the subgraph.

According to the definition of this subgraph, all nodes in the connected graph are included in the covering set, but only a fraction of the edges are selected. The remaining edges, in fact, define the connection operations between subgraphs. There is a special cover set S_0 where the number of nodes in $\forall G_i \in S_o$, G_1 is 1 and there are no edges in the subgraph. S_0 serves as the initial state for strategy selection in the query optimization of this paper.

For a subgraph G_1 in the covering set S, there is the following strategy based on its number of nodes. If subgraph G_1 has a node count of 1, no connection is defined. If the number of nodes of G_1 is 2, the default symmetric hash connection is used. Also. If the number of nodes of G_1 is greater than 2, a replicated connection is used for G_1. When the number of nodes of G_1 is greater than 2, low savings are bound to result.

In fact, it is possible to traverse all possible coverage sets by adaptively connecting subgraphs.

Definition 3: Given two subfigures G_i and G_j of Figure G_Q.

Set $e = (n_x, n_y)$ to be an edge of G_Q which satisfies $n_x \in G_i V, n_y \in G_j Y$. G_i and G_j can be connected by an edge e. The result of the connection is a new subgraph G_{ij} of which $G_i V = G_i V \bigcup G_j V, G_y.E = G_i.E \bigcup G_j.E \bigcup \{e\}$. By connecting the two subgraphs, a new graph is produced. For any two nodes in the graph, there are paths connecting these two points. The strategy selection method below illustrates how to traverse all possible cover sets by connecting graphs. The special covering set S_o serves as the initial state. After that, traverse all possible combinations of i edges selected from the graph. For a particular combination, a connection scheme temp can be generated, which is initialized to S_o. The selected edges are used to connect subgraphs in temp. Given a node and a connection scheme, the function getGraph returns the subgraph containing this node. The results of the coverage set are categorized in terms of candidate schemes. When all the schemes are generated, the connection scheme with minimum query overhead is selected as the connection scheme for this paper. The strategy selection algorithm searches for all possible schemes. Therefore, the complexity is estimated as follows:

$$\cos t = \sum_{i=1}^{C} \binom{C}{i} = 2^C - 1 \tag{3}$$

where $C = | G_Q.E |$. In most cases, very few relational tables are designed in a join and C is a very small value. This paper shows the effectiveness of the strategy selection algorithm in the experimental section below.

II. C. PGA-MMAS based multi-connection query optimization strategy

II. C. 1) PGA-MMAS Design Ideas

Firstly, GA initializes the initialization encoding of the final connection relationship, creates the initial population randomly, and selects the better population after selection, crossover, and mutation operations. Applying the adaptive dynamic fusion method proposed in this paper, the GA is made to transition to MMAS appropriately at the right time, the better population is transformed into the initialized pheromone distribution of MMAS, and the ants connect the relations according to the transfer probability, and update the pheromone matrix according to the improved MMAS pheromone updating rule, and the cycle iterates, and finally converges to obtain an optimal QEP.

II. C. 2) PGA-MMAS design process

The core elements of the genetic algorithm [40] include problem abstraction coding, initial population setting, fitness function design, genetic operation design and control parameter setting.

1) Coding and fitness function.

In this paper, for the characteristics of the distributed multi-table join query problem, string structured coding is adopted, and the chromosome is divided into two parts: the relation connection order coding segment and the relation size and corresponding site coding segment.

The goal of query optimization is to obtain the execution plan with the minimum connection cost.In GA, the cost of each QEP in each generation of population is calculated, and then the inverse of the required cost is taken as the value of the fitness function for each QEP, which is calculated as:

$$f(x) = 1 / COST(x) \tag{4}$$

The fitness function value is inversely proportional to the execution cost of the query plan, the higher the fitness function value, the better the corresponding QEP.

2) Population initialization [41].

A set of populations of a specific size is first randomly generated. Each individual in the initial population is generated using a heuristic rule to avoid Cartesian product operations in the individual, because two relations for Cartesian product operations will not only produce a large execution cost, the operation to obtain a relatively large intermediate results will also increase the cost of the subsequent connection.

3) Selection operator.

Selection is the most important operation in GA and is also the criterion for good or bad performance of GA. If the influence of the fitness on the survival probability of an individual is large in this process, then only a few better individuals will appear in the solution population, and thus fall into a local optimal situation. On the contrary, if the fitness is unable to hold the change of individual survival probability, it causes the algorithm to have no regular random wandering behavior and fails to converge. To avoid the above problems, the following selection operator is introduced in this paper.

Define all individuals within the current population as set P and $x^{(1)}, x^{(2)}, \cdots, x^{(n)}$ as a fixed permutation of set P. If x is an individual of P, denoting the fitness of that individual, the fitness of population P is defined:

$$s(P) = \sum_{i=1}^{n} f(x^{(i)}) \tag{5}$$

The relative fitness of any individual $x \in P$ is defined as $r(x) = f(x) / s(P)$. Relative fitness $r(x)$ captures the weight of the fitness of individual $x^{(0)}$ in the fitness of that population. The probability of selection is proportional to the individual fitness function. Cumulative fitness is defined as:

$$c(x^{(k)}) = \sum_{i=1}^{k} r(x^{(\theta)}) \tag{6}$$

Before making the selection, generate a random real number between 0 and 1. If t satisfies $r(x^{(k)}) \le t \le r(x^{(k+1)})$, select the $k+1$rd individual. Repeat the process until the selection of the parent of the next generation of the population is completed.

4) Crossover operator.

The crossover operation enables the search ability of GA to be improved by leaps and bounds. The method used in this paper is the inverted mutation method. Inversion operation refers to reversing the order of gene arrangement between two randomly specified loci in the coding strings of an individual, thus forming a new chromosome, i.e., generating a new linkage order.

The key to hybrid algorithms is to find the optimal articulation point of the two algorithms. In this paper, an adaptive dynamic fusion method is proposed to ensure that GA and MMAS converge at the best moment.

1) Set the minimum genetic iteration number and the maximum genetic selection number.

2) Count the evolution rate of the offspring population during GA iteration and set the minimum evolution rate of the offspring population.

3) Make a judgment within the set number of iterations, if the evolution rate of the subgeneration population in n consecutive generation is less than the minimum evolution rate, then it can be judged that the GA optimization efficiency is low at this time, and the GA process can be stopped and enter the MMAS.

Later generations have proposed a series of improvement measures for the problem that the ant colony algorithm has serious deficiencies in convergence speed, and the max-min ant colony system is an enhanced version of ACA.

1) Initialization of information cords.

The initial value of the pheromone of each path is set to the maximum value $\tau_{\max}$ in MMAS, and in order to solve the problem of the lack of pheromone guidance in the early stage of MMAS, in this paper, GA is applied to obtain a set of better solutions transformed into a certain amount of pheromone:

$$\tau_l = \tau_C + \tau_G \tag{7}$$

where, τ_C is the pheromone constant which is set according to the query optimization problem in this paper. τ_G is then the initialized pheromone transformed according to the optimized solution obtained by the genetic algorithm.

2) Transfer probability.

Each ant on the node is selected under according to the defined transfer probability:

$$P_{ij}^k = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{i \in allined_k} \tau_{is}(t)\eta_{is}(t)} & j \in allowed_k \\ 0 & \text{other} \end{cases} \tag{8}$$

3) Pheromone update.

Localized pheromone updating formula:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + (1-\rho)\tau_{ij}^{int} \tag{9}$$

During path search, the ants select which relation to connect next. The local pheromone is updated once for each ant connecting each relation.

MMAS selects the best ants after all ants finish connecting, and only the best ants passing through the path can leave pheromone. In order to speed up the algorithm evolution, this paper simultaneously cuts the pheromone on the worst path for global pheromone update.

The global pheromone update formula:

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + (1-\rho)\Delta\tau_{ij}^{beed} \tag{10}$$

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + (1-\rho)\Delta\tau_{ij}^{beed} \tag{11}$$

$$\Delta\tau_{ij}^{beed} = 1/L_{beed} \tag{12}$$

$$\Delta\tau_{ij}^{waxt} = 1/L_{waxt} \tag{13}$$

$\tau_i(t+n)$ represents the information cords on the connection edges of relation (i,j) in the optimal connection order after all ants have completed the connection of all relations in the query graph.

Represent the cost of the optimal and worst connection order during this iteration, respectively.

GA and MMAS are time-consuming when dealing with the multi-connection query problem for large-scale data tables, in order to address this shortcoming, this paper re-explored the application of hybrid genetic ant colony algorithms from the perspective of parallel and distributed processing.

Parallel GA-MASS takes advantage of distributed computing clusters by assigning a GAMASS program to several processors in parallel to increase the program execution speed and shorten the time required for algorithm execution.

In PGA-MMAS, the entire population is first divided into q equal-sized sub-populations, and each thread independently performs genetic operations on one sub-population. The following strategies are executed for each individual to be exchanged passed from other populations:

1) Randomly select one of the local to-be-evolved parents.

2) Compare the fitness of the local parent with that of the to-be-exchanged individual passed on from the other subpopulation and select the largest as the final parent.

During each evolutionary process, each sub-population in each retains its own local optimal solution to replace the worst solution in the next evolution. After each sub-population has completed the preset number of evolutions, the main thread normalizes the local optimal solutions of each sub-population to obtain the globally better solution of the whole population for output to MMAS.

Similarly, the optimization process of q ant colonies with equal number of ants is run in parallel at the same time. Until a global optimal QEP is obtained, the flow of the PGA-MMAS algorithm is shown in Fig. 1.

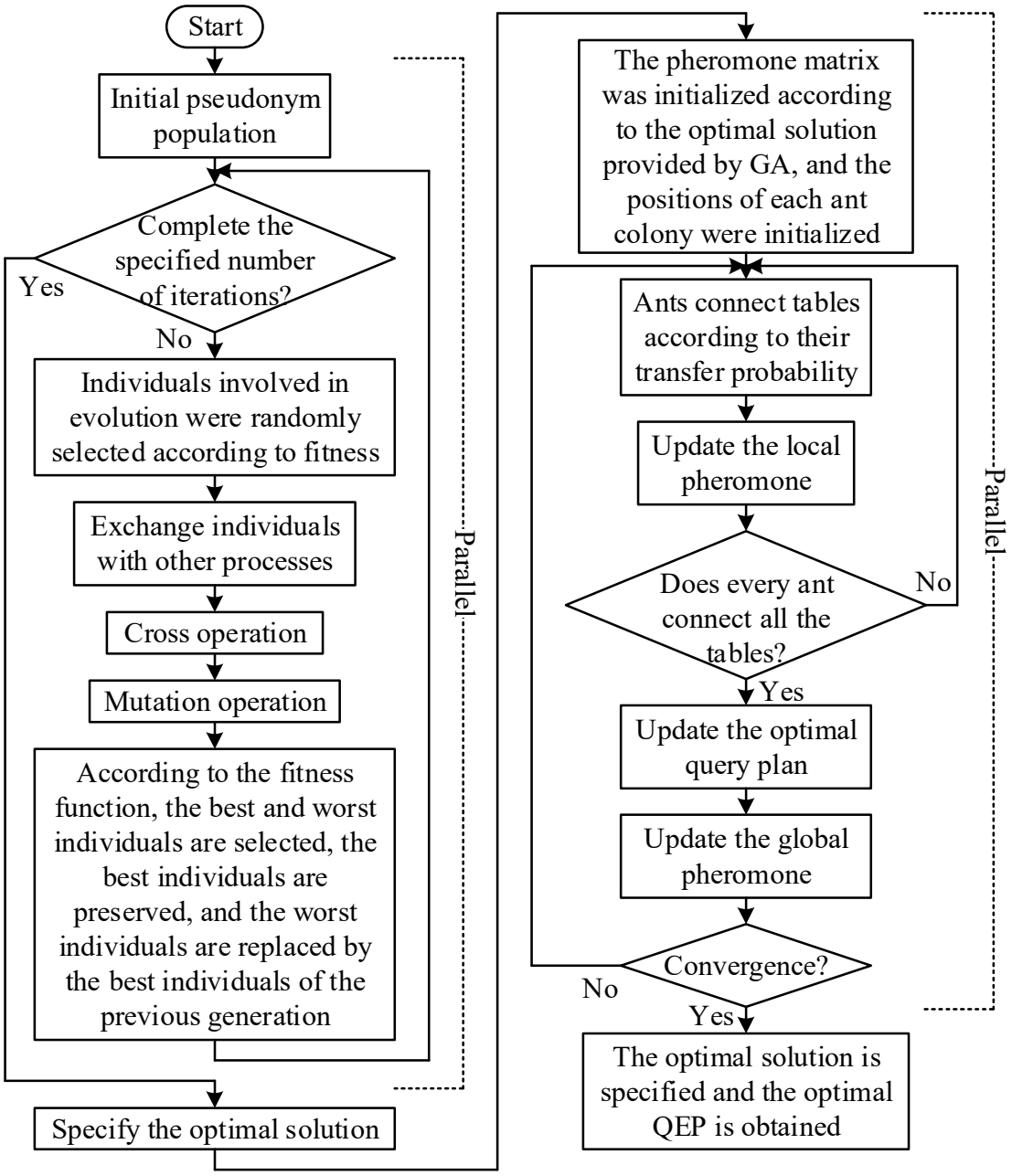

Figure 1: PGA-MMAS process

III. Query effectiveness analysis of distributed efficient query optimization algorithms

III. A. Performance evaluation of PGA-MMAS based algorithms

III. A. 1) Experimental design

In order to evaluate the performance of the improved genetic ant colony algorithm proposed in this paper, the performance of four algorithms, including single genetic algorithm, single ant colony algorithm, max-min ant colony algorithm, and the performance of the improved max-min ant colony algorithm based on parallel genetics proposed in this paper, were compared under the same experimental environment.

This paper divides the experimental process into three parts:

The first part is to verify the optimization effect of the improvement points of the parallel genetic-maximum minimum ant colony algorithm based on the parallel genetic-maximum ant colony algorithm respectively, which mainly includes the verification of the adaptive selection strategy, the verification of the crossover operation, the verification of the population iteration method, and this part mainly compares the optimization effect of the execution time, the number of convergent generations, and the execution cost of the algorithm's various improvement points.

The second part is designed to validate the single genetic algorithm, single ant colony algorithm, maximum minimum ant colony algorithm and parallel genetic-maximum minimum ant colony based algorithm under different number of relations, starting from the total execution time.

The third part is to analyze each of the four under the same number of relations and compare the convergence speed as well as the convergence cost.

III. A. 2) Experimental results

According to the experimental design, this paper validates the parallel genetic-maximum minimum ant colony algorithm based on the parallel genetic-maximum ant colony algorithm, first of all, this paper compares and validates each improvement point, in this experiment, in order to ensure the independence of each improvement point as well as not interfering with each other, it is tested separately. In the experimental environment, the experiment set the same parameters, in order to exclude chance, the algorithm runs 100 times and then take the average, the following results are obtained. Figure 2 shows the comparison results of the algorithms, where optimization point 1 represents the adaptive selection strategy, optimization point 2 represents the improved crossover strategy, and optimization point 3 represents the improved population iteration method. Each optimization point is better than the pre-improvement algorithm in terms of the number of convergence generations and execution time. Among them, the optimization effect of the improved crossover operation is more obvious than the other two, and the solution efficiency is improved by 10%.

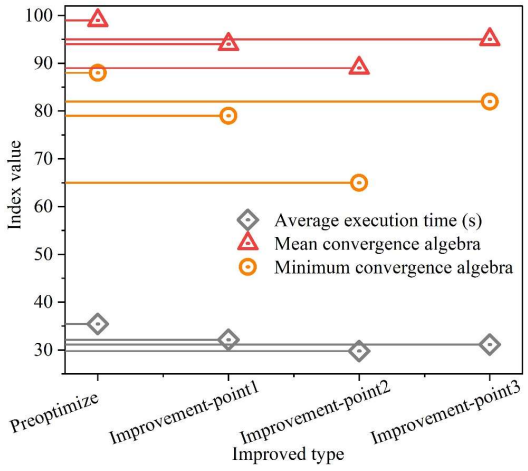

Figure 2: Comparisons of the results of improvement points

According to the second part of the experimental design, in the experimental environment, this paper sets the number of basic relational tables of the database as 1-10 respectively. Other experimental parameters are consistent with the above experiments, and each of the four algorithms: genetic algorithm, ant colony algorithm, max-min ant colony algorithm and improved max-min ant colony algorithm are executed 100 times, and then the execution time is averaged for comparison, and the results of the relationship between the number of database relations and the total execution time are shown in Figure 3.

As can be seen from the figure, at the number of database relational table connections 1-3, the execution time of the algorithms does not differ much, and as the number gradually increases, the optimized max-min ACO algorithm has significantly the shortest total execution time compared to the other algorithms. This is because the algorithm improves the diversity of the population while increasing the probability of the optimal solution, improves the optimization efficiency, enhances the global convergence, and thus shortens the optimization search time, and when the number of database relational tables is 10, the optimized max-min ACO algorithm has a total execution time of only 75s.

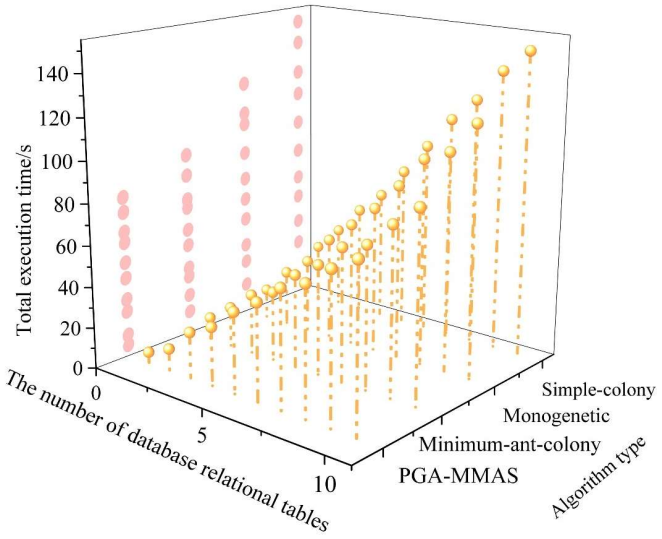

Figure 3: Number of database relationships and total execution time

In order to verify the trend of the cost of the improved algorithms with the number of iterations when the number of basic relational tables of the database is constant. In this paper, the number of database basic relational tables is set to 6, and then the performance of four algorithms is verified, and the trend of the execution cost of each algorithm is shown in Figure 4. Compared with the pre-optimization algorithm, the optimized algorithm converges faster, and the total cost tends to level off at 25 iterations, while the pre-optimization algorithm tends to approximate level off at 35 iterations and the cost is higher than the optimized algorithm. Although the single genetic algorithm converges quickly, it tends to level off at 45 iterations, and the final cost is higher than the first two, appearing to be locally optimal. From the figure, it can be seen that the optimized algorithm outperforms the other three algorithms.

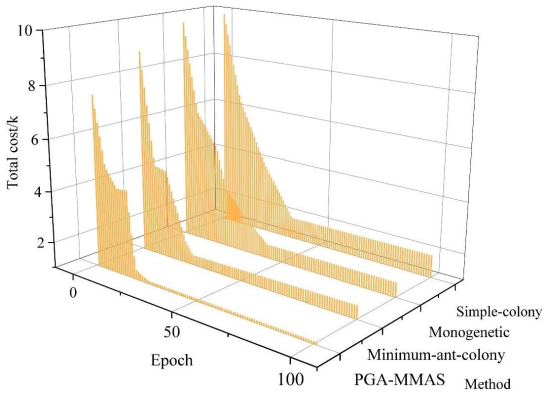

Figure 4: Iterative process and cost

III. B. Database queries based on PGA-MMAS

III. B. 1) Experimental setup

In order to verify the effectiveness of the PGA-MMAS based optimization method, the method is used to index and query a distributed intelligent transportation database in a region. The server nodes in this database exist in topological form. The database mainly records the basic information of all vehicles passing through each road traffic

intersection, including vehicle model, license plate number, vehicle color, recording time, traffic intersection number, lane number, near view picture and far view picture. The index query is for all white vehicles passing through traffic intersection No. 1 on August 10, 2024 from 9:30 to 9:40. The relevant parameters of the experiment are set as follows: the number of ant colonies is 2, the size is 30, the pheromone importance weight is 2.5, the heuristic information importance weight is 3.5, the pheromone volatilization factor of colony 1 is 0.15, the pheromone volatility factor of colony 2 is 0.1, the learning operator is 0.08, the maximum number of iterations is 100, the number of iterations at which the colony starts to use the pheromone smoothing strategy is 30, the number of iterations at which pheromone communication between the colonies begins is 30 and the number of iterations at which pheromone exchange between the colonies is not allowed. The maximum number of iterations is 100, the number of iterations at which the colony starts to use the pheromone smoothing strategy is 30, and the number of iterations at which the colony starts to communicate pheromone among themselves is 30.

III. B. 2) Experimental results

The index query situation is shown in Fig. 5, the index query command is issued by server node 0, and the index query information is distributed on server nodes 1,4,5,8,13,15,24. The index query path is the optimal path through the server node where the index query content is located, which indicates that the optimization method based on PGA-MMAS is effective.

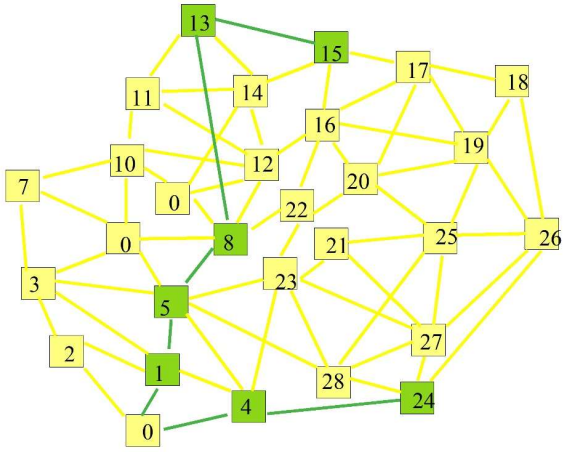

Figure 5: Distributed efficient query results

In order to verify the index query load balancing ability of PGA-MMAS based optimization method in distributed database, the experiment creates 8 groups of different sizes of user index query service requests, and at the same time issues index query requests to the intelligent traffic database. The index query load balancing degree is used as the evaluation index, the smaller the load balancing degree is, the stronger the load balancing ability is. The number of user index query requests are set to 1-2500, the feedback interval is 5s, and the test results are shown in Figure 6.

Before the load balancing operation on the distributed database, the index query load distribution of the database is very unbalanced, the maximum load balancing degree reaches 6.5%, and the fluctuation is large. After adjusting the distributed database index query load through the optimization method based on PGA-MMAS, the load balance degree has decreased significantly, the maximum load balance degree is 1.6%, and the overall curve trend is smooth. This shows that the optimization method based on PGA-MMAS achieves the index query load balance of distributed database.

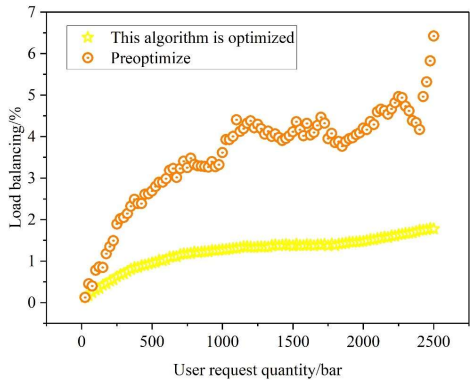

Figure 6: This paper compares the performance of the index query load

Response time, refers to the server's response time to the user's application. It is an important indicator of the performance of the database system, but also from the side to reflect the size of the operating costs. The following optimization method based on PGA-MMAS is used to perform index query operations on distributed databases with different data volumes, and analyze the index query performance of the method with different index query load balancing degrees through the response time metrics. The data volume of the distributed database is set to be 100,000-2,000,000, and the response time of the query is shown in Figure 7.

The optimization method based on PGA-MMAS has a better performance of database index query response time when processing a large amount of data under different index query load balance. When the amount of data in the distributed database is less than 1 million, the response time of the three index query load balancing degree is not more than 50 ms. When the amount of data in the distributed database reaches 1.5 million, the response time of the three index query load balancing degree is not more than 55 ms, especially when the load balancing degree is 1%, the response time of the whole process is less than 33 ms. This shows that the optimization method based on PGA-MMAS can efficiently complete the index query for distributed databases with different data volumes.

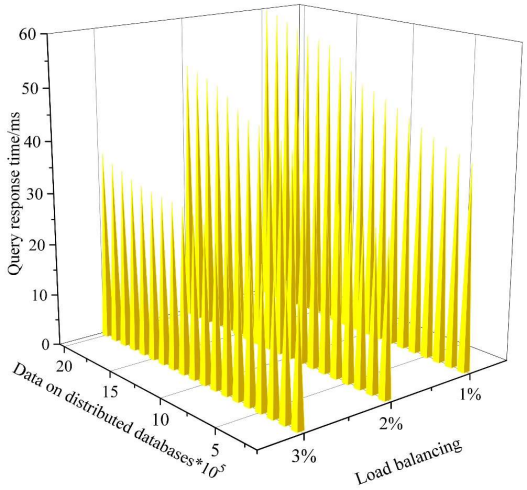

Figure 7: The database index query response time of different scale data

III. C. Performance testing of PGA-MMAS based query optimization

This subsection focuses on comparing the query latency of SparkSQL with the distributed database query method based on the PGA-MMAS based design described in this paper under various types of SQL queries. The tests are repeated multiple times for each SQL statement to ensure the accuracy of the results. The query latency test is divided into two categories, one is to verify the query latency of the execution plan after splitting of a certain class of operators by testing multiple simple statements, and the other is to verify the query latency of the plan after splitting under the combination of multiple operators by testing complex statements.

III. C. 1) Single Table Scan Query Latency Testing

The query latency for 10 queries is shown in Table 1. The data in the table shows that the single table scanning latency of SpakSQL is more than 10 times of the distributed database querying method designed in this paper based on PGA-MMAS, and the average latency of the distributed database querying method designed in this paper based on PGA-MMAS is reduced by 91.55% compared to SpakSQL.

Table 1: Table scan query delay test data

Query serial number	SparkSQL(s)	This method(s)
1	21.472	1.832
2	22.631	1.843
3	21.844	1.919
4	21.138	1.976
5	22.849	1.787
6	21.332	1.796
7	21.593	1.892
8	22.013	1.804
9	21.729	1.858
10	21.634	1.744
Mean	21.824	1.845

III. C. 2) Multi-table join query latency testing

The query latency of 10 multi-table join queries is shown in Table 2. The data in the table shows that the multi-table join latency of SpakSQL is more than three times of the distributed database query method designed based on PGA-MMAS in this paper. Compared with SparkSQL, this paper's distributed database query method designed based on PGA-MMAS has a higher degree of parallelism, and the multi-table join query latency of this paper's method is reduced by 71.94% compared to the SpakSQL method.

Table 2: Multiple table connection query delay test data

Query serial number	SparkSQL(s)	This method(s)
1	46.055	13.126
2	45.502	12.537
3	45.610	13.182
4	46.107	12.540
5	46.162	12.931
6	46.036	12.972
7	46.135	12.978
8	45.916	13.261
9	45.741	12.652
10	46.343	12.775
Mean	45.961	12.895

III. C. 3) Grouped Aggregate Query Latency Testing

The query latency of grouped aggregation for 10 queries is shown in Table 3. The data in the table shows that the grouped aggregation query latency of SpakSQL is more than 12 times higher than the distributed database query method designed based on PGA-MMAS in this paper, and the overall query latency time mean of this paper's method is reduced by 91.91% compared to the comparison method.

Table 3: Group query delay test data

Query serial number	SparkSQL(s)	This method(s)
1	175.618	14.36
2	176.047	14.512
3	176.202	14.908
4	176.787	14.252

5	176.522	14.198
6	175.912	13.634
7	176.252	13.603
8	175.731	13.796
9	175.689	15.018
10	176.373	14.110
Mean	176.113	14.239

III. C. 4) Sort Query Latency Testing

The query latency of 10 queries for the sort query latency test is shown in Table 4, through the data in the table we can see that the grouped aggregation query latency of SpakSQL is more than 5 times than that of this paper's distributed database querying method based on the design of PGA-MMAS, and the average value of sort query latency of this paper's method is reduced by 80.98% compared to the comparison method.

Table 4: Sort query delayed test data

Query serial number	SparkSQL(s)	This method(s)
1	19.384	4.221
2	19.182	3.421
3	17.846	3.324
4	18.755	3.332
5	19.364	4.049
6	18.514	4.011
7	19.412	3.731
8	19.133	2.725
9	18.752	4.074
10	17.854	2.904
Mean	18.820	3.579

III. C. 5) Complex Query Latency Testing

Complex Query Latency The query latency for 10 queries is shown in Table 5. The data in the table shows that the complex query latency of SpakSQL is more than 4 times of the distributed database query method designed based on PGA-MMAS in this paper, and the mean value of complex query latency of this paper's method is reduced by 76.82% compared with the comparison method. In the testing process, the computational performance of the distributed database query method designed based on PGA-MMAS in this paper is fully utilized to improve the computational efficiency.

Table 5: Complex query delay test

Query serial number	SparkSQL(s)	This method(s)
1	231.31	53.237
2	230.825	54.086
3	232.136	53.565
4	231.812	53.412
5	231.306	52.853
6	230.701	52.547
7	230.78	54.037
8	231.72	54.289
9	232.23	54.473
10	232.029	54.057
Mean	231.485	53.656

IV. Conclusion

In this paper, a distributed database index query method based on PGA-MMAS optimization algorithm is designed for the current distributed database query optimization algorithm with low query efficiency.

(1) The optimization algorithm based on PGA-MMAS outperforms the comparison algorithms such as single genetic algorithm, single ant colony algorithm and maximum-minimum ant colony algorithm in terms of convergence, execution time and probability of optimal solution, and the improved crossover operation improves the solution efficiency by 10%, and the total execution time is also greatly reduced compared with the comparison algorithm, which improves the operation efficiency of the algorithm while maintaining the final cost at a lower level. It proves that the improvement strategy of the PGA-MMAS-based optimization algorithm is effective and optimizes the performance of the algorithm.

(2) In the example of database query, this paper's algorithm queries the optimal path of the server node, and the overall curve of the algorithm has a smooth trend, and when the number of user requests is 2500, the maximum load balance is only the most 1.6%, and the whole response time is less than 40ms. it shows that this paper's method can effectively optimize the load distribution of the distributed database, and it has an excellent database query capability.

(3) From the performance of the distributed query optimizer, the database using the query optimizer designed based on this paper's PGA-MMAS algorithm has a different degree of query latency reduction compared to SparkSQL in the areas of single-table scanning, multi-table joins, grouped aggregation, sorting, and complex query latency, with a degree of reduction of more than 60%.

References

[1] Bergman, S., Asplund, M., & Nadjm‐Tehrani, S. (2020). Permissioned blockchains and distributed databases: A performance study. Concurrency and Computation: Practice and Experience, 32(12), e5227.

[2] Kumar, D., & Jha, V. K. (2022). A review on recent trends in query processing and optimization in big data. Wireless Personal Communications, 1-22.

[3] Nashat, D., & Amer, A. A. (2018). A comprehensive taxonomy of fragmentation and allocation techniques in distributed database design. ACM Computing Surveys (CSUR), 51(1), 1-25.

[4] Fuaad, H. A., Ibrahim, A. A., Majed, A., & Asem, A. (2018). A survey on distributed database fragmentation allocation and replication algorithms. Current Journal of Applied Science and Technology, 27(2), 1-12.

[5] Jafarinejad, M., & Amini, M. (2018). Multi-join query optimization in bucket-based encrypted databases using an enhanced ant colony optimization algorithm. Distributed and Parallel Databases, 36, 399-441.

[6] Thokala, V. S. (2021). A Comparative Study of Data Integrity and Redundancy in Distributed Databases for Web Applications. Int. J. Res. Anal. Rev, 8(4), 383-389.

[7] Ueda, K., & Oguchi, N. (2022). Highly Reliable Redundant Data Allocation Method for High Availability Distributed Clusters. ITE Transactions on Media Technology and Applications, 10(4), 190-197.

[8] Qasim, A., Ghouri, A., & Munawar, A. (2024). An effective approach for reducing data redundancy in multi-agent system communication. Multiagent and Grid Systems, 20(1), 69-88.

[9] Azhir, E., Navimipour, N. J., Hosseinzadeh, M., Sharifi, A., & Darwesh, A. (2019). Query optimization mechanisms in the cloud environments: A systematic study. International Journal of Communication Systems, 32(8), e3940.

[10] Bachhav, A., Kharat, V., & Shelar, M. (2018). Novel architecture of an intelligent query optimizer for distributed database in cloud environment. Journal of Advanced Database Management & Systems, 5(2), 28-32.

[11] Yadav, P. K., & Rizvi, S. (2018). Query Optimization: Issues and Challenges in Mining of Distributed Data. Big Data Analytics: Proceedings of CSI 2015, 693-698.

[12] Rana, I. A., Aslam, S., Sarfraz, M. S., & Shoaib, U. (2018). Analysis of Query Optimization Components in Distributed Database. Indian Journal of Science and Technology, 11(18), 10-17485.

[13] ATTRIBUTE, M. (2019). QUERY OPTIMIZATION ON DISTRIBUTED HEALTH DATABASE DBD FOR SUPPORTING DATA CENTER WITH MATERIALIZED VIEW AND MINIMIZING ATTRIBUTE INVOLVEMENT. Journal of Theoretical and Applied Information Technology, 97(11).

[14] Li, X., Yu, H., Yuan, L., & Qin, X. (2022). Query optimization for distributed spatio-temporal sensing data processing. Sensors, 22(5), 1748.

[15] Ramu, V. B. (2023). Optimizing Database Performance: Strategies for Efficient Query Execution and Resource Utilization. International Journal of Computer Trends and Technology, 71(7), 15-21.

[16] Boicea, A., Radulescu, F., Truica, C. O., & Urse, L. (2016). Improving Query Performance in Distributed Database. Journal of Control Engineering and Applied Informatics, 18(2), 57-64.

[17] Gadde, H. (2024). Optimizing Transactional Integrity with AI in Distributed Database Systems. International Journal of Advanced Engineering Technologies and Innovations, 1(2), 621-649.

[18] Muzammal, M., Qu, Q., & Nasrulin, B. (2019). Renovating blockchain with distributed databases: An open source system. Future generation computer systems, 90, 105-117.

[19] Amer, A. A., Sewisy, A. A., & Elgendy, T. M. (2017). An optimized approach for simultaneous horizontal data fragmentation and allocation in Distributed Database Systems (DDBSs). Heliyon, 3(12).

[20] Ye, S., & Peng, Y. (2018). An Optimization for Distributed Database Multi-join Query Based on Improved Genetic Algorithm. DEStech Transactions on Computer Science and Engineering, 10.

[21] GAO, J., HAN, Y., LIN, Y., MIAO, H., & XU, M. (2024). Learned Distributed Query Optimizer: Architecture and Challenges. ZTE Communications, 22(2), 49.

[22] Tang, M., Yu, Y., Mahmood, A. R., Malluhi, Q. M., Ouzzani, M., & Aref, W. G. (2020). Locationspark: In-memory distributed spatial query processing and optimization. Frontiers in big Data, 3, 30.

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

[24] Gao, J., Liu, W., Li, Z., Zhang, J., & Shen, L. (2020). A general fragments allocation method for join query in distributed database. Information Sciences, 512, 1249-1263.
[25] Li, A., Wang, X., Wang, X., & Bohan, L. (2020). An improved distributed query for large-scale RDF data. Journal on Big Data, 2(4), 157.
[26] Abdalla, M. H., & Karabatak, M. (2020, June). To review and compare evolutionary algorithms in optimization of distributed database query. In 2020 8th International Symposium on Digital Forensics and Security (ISDFS) (pp. 1-5). IEEE.
[27] Lakshmi, S. V., & Vatsavayi, V. K. (2017). TEACHER-LEARNER & MULTI-OBJECTIVE GENETIC ALGORITHM BASED QUERY OPTIMIZATION APPROACH FOR HETEROGENEOUS DISTRIBUTED DATABASE SYSTEMS. Journal of Theoretical & Applied Information Technology, 95(8).
[28] Padia, S., Khulge, S., Gupta, A., & Khadilikar, P. (2015). Query optimization strategies in distributed databases. International Journal of Computer Science and Information Technologies, 6.
[29] LIU, X., & DONG, Z. (2024). Distributed Database Index Query Optimization Method Based on Consistent Hash Algorithm. Journal of Jishou University (Natural Sciences Edition), 45(1), 36.
[30] Sharma, M., Singh, G., & Singh, R. (2019). A review of different cost-based distributed query optimizers. Progress in Artificial Intelligence, 8, 45-62.
[31] Mohsin, S. A., Darwish, S. M., & Younes, A. (2021). Qiaco: a quantum dynamic cost ant system for query optimization in distributed database. IEEE Access, 9, 15833-15846.
[32] Du, Y., Cai, Z., & Ding, Z. (2024). Query Optimization in Distributed Database Based on Improved Artificial Bee Colony Algorithm. Applied Sciences, 14(2), 846.
[33] Panahi, V., & Navimipour, N. J. (2019). Join query optimization in the distributed database system using an artificial bee colony algorithm and genetic operators. Concurrency and Computation: Practice and Experience, 31(17), e5218.
[34] Kaseb, M. R., Haytamy, S. S., & badry, R. M. (2021). Distributed query optimization strategies for cloud environment. Journal of Data, Information and Management, 3(4), 271-279.
[35] Zheng, B., Li, X., Tian, Z., & Meng, L. (2022). Optimization method for distributed database query based on an adaptive double entropy genetic algorithm. IEEE Access, 10, 4640-4648.
[36] Mohsin, S. A., Younes, A., & Darwish, S. M. (2021). Dynamic cost ant colony algorithm to optimize query for distributed database based on quantum-inspired approach. Symmetry, 13(1), 70.
[37] Azhir, E., Navimipour, N. J., Hosseinzadeh, M., Sharifi, A., Unal, M., & Darwesh, A. (2022). Join queries optimization in the distributed databases using a hybrid multi-objective algorithm. Cluster Computing, 1-16.
[38] Byung-Jung Oh,Soo-Min Ahn & Kyung-Chang Kim. (2011). Performance Comparison of Column-Oriented and Row-Oriented Database Systems for Star Schema Join Processing. Journal of the Korea Society of Computer and Information,16(8),29-38.
[39] Bansal Rohit,Kumar Deepak & Kumar Sushil. (2020). Multi-objective Multi-Join Query Optimization using Modified Grey Wolf Optimization. International Journal of Advanced Intelligence Paradigms,17(1/2),1-1.
[40] Michelle Setiyanti,Genrawan Hoendarto & Jimmy Tjen. (2025). Enhancing Water Potability Identification through Random Forest Regression and Genetic Algorithm Optimization. Engineering Headway,18,101-110.
[41] Zhidan Li,Wei Liu,Hongying Zhao & Wenjing Pu. (2025). Dung beetle optimization with composite population initialization and multi-strategy learning for multi-level threshold image segmentation. Signal, Image and Video Processing,19(3),255-255.