

Received November 20, 2021, accepted January 3, 2022, date of publication January 7, 2022, date of current version January 13, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3141589

Optimization Method for Distributed Database Query Based on an Adaptive Double Entropy Genetic Algorithm

BINGXU ZHENG¹, XIANG LI², ZHENZHEN TIAN¹, AND LIMIN MENG¹

¹College of Information Engineering, Zhejiang University of Technology, Hangzhou 310000, China

²Zhongtong Fuhuizhan Technology Company, Hangzhou 310000, China

Corresponding authors: Bingxu Zheng (zbx163email@163.com) and Xiang Li (15305719889@189.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61871349; and in part by the Natural Science Foundation of Zhejiang Province under Grant LQ19F010013 and Grant LY18F010024.

ABSTRACT In a distributed database environment, multi-join query optimization is one of the key factors affecting database performance. Genetic algorithms have a good application in dealing with this type of problem. However, the traditional genetic algorithm has the problems of low efficiency and easily falls into the precocity when dealing with query optimization, which is mainly caused by the lack of population diversity. Therefore, this paper sets up a mathematical model for distributed database query optimization and proposes an adaptive genetic algorithm based on double entropy. We introduced a genetic algorithm with two types of entropy: genotype and phenotype. Genotype entropy was used to optimize the distribution of the initial population, ensuring that the initial population has good population diversity. Phenotype entropy is used to optimize the genetic strategy, which can be divided into individual entropy and population entropy. Individual entropy is used to optimize the selection strategy, and population entropy is used to optimize the crossover and mutation operators to maintain the population diversity in the iteration process and accelerate the speed of iteration. The experimental results show that the algorithm proposed in this paper is effective for query optimization of a distributed database.

INDEX TERMS Database, distributed, multi-join query, genetic algorithm, entropy.

I. INTRODUCTION

In the era of big data, in the face of increasing mass data, the disadvantages of traditional centralized database are increasingly appearing. To adapt to complex and changeable network requirements and massive data, the distributed database system was born at a historic moment. A distributed database system (DDBS) is a collection of data that is logically related to each other but distributed on different sites of the computer network [1]. These data can not only be run separately, but also communicate with each other through the computer network, and respond to a complex task together to form a uniform whole. The performance of DDBS depends on its ability to handle query requirements in an efficient manner, and the query processing in DDBS needs to transfer data between different sites on the network. In DDBS, the query cost mainly includes CPU, I/O, and communication costs, and communication cost is the most important factor affecting the

performance of a query. The communication cost is the cost of transferring data among the different sites that participate in the query. Data transmission and local data processing constitute the distributed query strategy, which is known as the query execution plan (QEP). Multi-join query is one of the most common operations in a distributed database. When multiple relationships are connected, there are many different orders for the same query, and each order corresponds to a QEP. As the number of relational tables increases, the number of different QEPs increases exponentially, which leads to high computational complexity. Therefore, the traditional database query method is inefficient in dealing with the query of massive data, and it is difficult to adapt to distributed queries. Therefore, seeking an intelligent method to quickly find the best QEP with the lowest communication cost among all QEPs in the search space, to reduce the query cost as much as possible and improve the efficiency of query response has become the focus of current research.

Scholars at home and abroad have proposed various strategies for the optimization of distributed database

The associate editor coordinating the review of this manuscript and approving it for publication was Jagdish Chand Bansal.

queries. Examples include SDD-1 [2], [3], dynamic programming [4], [5], simulated annealing [6], [7], genetic algorithm [8]–[10] and so on. Paper [11] proposed a query optimization method based on the Tabu-GEP algorithm, which combines the Tabu search strategy with the GEP algorithm. It improves the performance of the classic GEP algorithm. The query time and generation time of the optimal query strategy were both significantly reduced compared to the original. However, the time complexity of the algorithm was still high. Paper [12] proposed an adaptive genetic algorithm, which reintroduced individuals scattered outside the convergence part into genetic operations and adaptively adjusted the evolutionary strategy according to the different fitness of individuals to maintain the diversity of individuals. However, it may also introduce undesirable genes, which slows down the optimization process. Paper [13] combined multiple ant colonies with genetic algorithm, overcoming the blindness of the early search of ant colony algorithm, and used the smooth mechanism and the mechanism of learning from each other among ant colonies to avoid falling into local optimum and precocity. It performs better in preventing the algorithm from falling into a local optimum and can obtain a better query strategy. However, the quality of the initial pheromone too depends on the results of the genetic algorithm. The work in [14] provided an HMSST+ algorithm to optimize the storage and query strategy of a distributed memory database. It uses an SST connection selection strategy to quickly calculate the optimal connection scheme. This algorithm can improve the query efficiency and has strong scalability. However, the improvement effect is not significant for more complex query statements.

Because the classical genetic algorithm is prone to prematurity, it is difficult to obtain an ideal optimal solution. In this paper, we introduced the concept of information entropy into the genetic algorithm and proposed an adaptive double-entropy genetic algorithm (ADEGA), which is based on two types of entropy. We used the genotype entropy of the population to optimize the initial population distribution. During the process of evolution, we selected an appropriate evolutionary strategy according to population phenotype entropy to adaptively adjust the genetic operator and maintain individual diversity in the evolution process. This can improve the global search ability of the entire algorithm and quickly obtain the optimal solution. Experiments show that the ADEGA algorithm can obtain good optimization results and effectively improve the efficiency of distributed database queries.

II. QUERY EXECUTION COST MODEL

A. THE REPRESENTATION OF QUERY EXECUTION PLAN

As the uncertainty of the join order of relational tables constitutes the diversity of QEPs, the QEPs of a distributed database can be represented by a query binary tree [15], as shown in Fig. 1. In the figure, the leaf nodes of the binary tree represent the relational tables in the database, and the intermediate nodes represent the intermediate result sets for the joins of

the left and right relational tables. In general, for a join of n relational tables, there are $n!$ different kinds of QEP. With the increase in the number of relational tables, the number of QEPs increases exponentially, which is similar to the classical TSP [16] problem, both of which are NP-hard problems. Therefore, it is almost impossible to search for the optimal QEP using an exhaustive method.

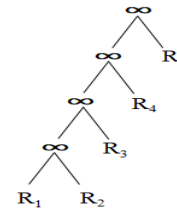


FIGURE 1. Query binary tree.

B. COST ESTIMATION

In this paper, we considered the multi-join query. Therefore, we use “∞” to represent the join operation of the two relational tables. For the join of two relational tables in the multi-join, $S = R_1 \infty R_2$, the record number of the intermediate result set after the join is

$$|S| = \frac{|R_1| \times |R_2|}{\prod_{c_i \in C} \max(V(c_i, R_1), V(c_i, R_2))} \quad (1)$$

where $|R_1|$ and $|R_2|$ represent the cardinality of table R_1 and table R_2 , which refer to the number of records of different tuples. C is the set of common attributes for the two relational tables that participate in the join operation. $V(c_i, R_1)$ and $V(c_i, R_2)$ represent the number of different values of the i th public attribute c_i in table R_1 and table R_2 . S represents the intermediate dataset after table R_1 and table R_2 are joined, and $|S|$ represents the cardinality of this intermediate dataset.

To estimate the size of the intermediate dataset after the join operation, we made two assumptions [17]. First, the attribute values were evenly distributed across the tuples of the table. Second, the values of the different attributes are independent. Therefore, the width W_R of table R can be expressed as:

$$W_R = \sum_{i=1}^n W_i \quad (2)$$

In the expression, i represents the i th attribute of table R , n represents the total number of attributes for table R , W_i is the width of i th attribute. The join attributes involved in the join operation can be divided into two types. If the join attribute is not a pure join attribute, the repeated join attributes should be removed from the result set, and only one join attribute can be retained. If the join attribute is a pure join attribute, all of them are removed from the result set. Therefore, the width W_S of the intermediate result set S formed by each join operation

can be expressed as

$$W_S = \begin{cases} W_{R_1} + W_{R_2} - 2W_{join(R_1, R_2)}, & \text{join}(R_1, R_2) \\ & \text{is pure join attribute} \\ W_{R_1} + W_{R_2} - W_{join(R_1, R_2)}, & \text{join}(R_1, R_2) \text{ is not} \\ & \text{pure join attribute} \end{cases} \quad (3)$$

$join(R_1, R_2)$ represents the join attributes of table R_1 and table R_2 .

Therefore, the size of the intermediate dataset, $Size(S)$, formed by a join operation is:

$$Size(S) = |S| \times W_S \quad (4)$$

Because the data of the distributed database are stored separately on data tables at different sites, table R_1 and table R_2 may be on the same site or on different sites. When they are not in the same site, the data of the smaller relational table must be transferred to the site on which the larger relational table is located. Therefore, it is necessary to calculate the transmission cost of the data among the sites. For the joining of two relational tables, $J = R_1 \circ R_2$, the cost model $cost(j)$ is given by

$$cost(j) = \begin{cases} \frac{|R_1| \times |R_2|}{\prod_{c_i \in C} \max(V(c_i, R_1), V(c_i, R_2))} \times W_j & R_1 \text{ and } R_2 \text{ are in the same site} \\ \frac{|R_1| \times |R_2|}{\prod_{c_i \in C} \max(V(c_i, R_1), V(c_i, R_2))} \times W_j + \min(Size(R_1), Size(R_2)) & R_1 \text{ and } R_2 \text{ are in the different sites} \end{cases} \quad (5)$$

In the expression, $\min(Size(R_1), Size(R_2))$ is the smaller value between table R_1 and table R_2 , W_j is the width of the intermediate dataset after the join. The upper part of equation (5) indicates that when two tables are joined at the same site, the cost only includes one part, that is, the cost of the join of two tables, because there is only the data join within the same site but no data transmission between different sites. The lower part of equation (5) indicates that when two tables are in different sites, the data need to be transferred between different sites, it produced additional transmission cost, so the cost includes two parts, the first part is the cost of join of tables, the second part is the cost of data transmission between different sites and the value is the smaller one between two tables. For a join query with n relational tables, the intermediate nodes are (j_1, j_2, \dots, j_n) , and the total cost estimation model, $COST$, is defined as

$$COST = \sum_{i=1}^{n-1} cost(j_i) \quad (6)$$

C. DATA DICTIONARY AND NETWORK PERFORMANCE MATRIX

1) GLOBAL DATA DICTIONARY

The global data dictionary [18] is a record of the global structure and information of a distributed database. It mainly includes some static information of data tables, such as the relevant information of fields, table names, number of table records, number of sites where the tables are located, and the database to which the table belongs. When a global data dictionary is being built, multiple records should be created if a field appears in more than one table, or if the table containing a field is distributed in more than one site.

In the process of distributed query, by accessing the global data dictionary, the relevant database information and site information can be obtained, so that the next query decomposition operation can be carried out.

2) NETWORK PERFORMANCE MATRIX

The network performance matrix is an intuitive representation of network performance between sites. The number of rows and columns is equal to the number of sites; thus, theoretically, it is a symmetric matrix. When cost evaluation involves data transmission between different sites, it is necessary to access the network performance matrix to obtain network performance parameters between sites and take these parameters into account in the cost evaluation. Assume that the network performance matrix M of the four sites is expressed as follows:

$$M = \begin{bmatrix} -1 & 1.2 & 3.2 & 1.8 \\ 1.2 & -1 & 2.5 & 0 \\ 3.2 & 2.5 & -1 & 1.5 \\ 1.8 & 0 & 1.5 & -1 \end{bmatrix}$$

From this performance matrix, we can find that the diagonal element represents the site communicates with itself, because there is no network communication cost when the site communicates itself, so the site's self-network performance is -1 . The other elements indicate that the network performance varies among different sites. The best performance was between site 1 and site 3, with a value of 3.2. The performance between site 1 and site 2 was the worst, with a value of 1.2. The value of 0 between site 2 and site 4 indicates that these two sites cannot communicate with each other owing to some faults.

III. IDEA OF OPTIMIZATION ALGORITHM

The genetic algorithm (GA) is a random search method derived from the evolution law of biology [19]. It exhibits strong robustness and fast convergence. The steps generally include selection, crossover, mutation, and others. However, classic genetic algorithms generally exist the precocity phenomenon, and it easily falls into the local optimum, which greatly affects the optimization ability of the algorithm. The main reason is that the search process is limited to a piece of area due to the lack of population diversity, and the results obtained are only the optimal solutions within this

area, rather than the global optimal solutions. Therefore, maintaining population diversity is very important for genetic algorithms, and many researchers have proposed various measures [20]–[22].

Entropy [23] is a quantitative index used to measure the diversity and richness of a system state. By monitoring and controlling the change in entropy, the system can change in a certain direction. Therefore, we introduced information entropy into the genetic algorithm [23], [24] as a control index of state change in the population, so that the algorithm can always maintain a good population diversity in the evolution process. In this paper, we propose an adaptive double-entropy genetic algorithm (ADEGA). Two types of entropy are used to control the algorithm process. Genotype entropy is used to optimize the generation of the initial population, so that the initial population has a better distribution and improves the initial genetic advantage of the population. Phenotype entropy is used to control the change in genetic operators during the evolution process. Phenotype entropy can be divided into population entropy and individual entropy. Population entropy mainly affects the selection process, whereas individual entropy acts on the crossover, mutation, and recombination processes. With these two types of entropy, the algorithm can adaptively adjust the evolutionary strategy, maintain population diversity, and obtain excellent optimization results.

The flow of the optimization algorithm designed in this paper is shown in Fig. 2.

IV. QUERY OPTIMIZATION BASED ON DOUBLE ENTROPY GENETIC ALGORITHM

Aiming at the query optimization problem of distributed database and the shortcoming of genetic algorithm, this paper proposed an improved algorithm that uses two types of entropy to optimize the genetic algorithm and applied it to the query of distributed database. The main content of the optimization algorithm in this paper includes the encoding of problem, fitness function, initial population optimization, genetic operators and so on.

A. ENCODING SCHEME AND FITNESS FUNCTION

In this paper, we chose real encoding to encode the tree structure of the query execution plan. It is assumed that each data table is encoded into a one-digit integer, and the order of the code string represents the access order of the data tables. For example, for code string 12345, the corresponding access order is 1→2→3→4→5. For the encoding of n tables, the encoding length is simply the number of tables.

The specific encoding method was as follows: First, we numbered the n data tables participating in the query from 1 to n , then encoded the leaf nodes of the binary tree into an ordered sequence from bottom to top according to the principle of post-order traversal, and the length of the sequence is n . Then, the access order of each site during the query process can be obtained according to the site to which each relational table belongs. Finally, the query cost can be obtained by substituting the relative data of tables and sites into the cost calculation model. For example, the

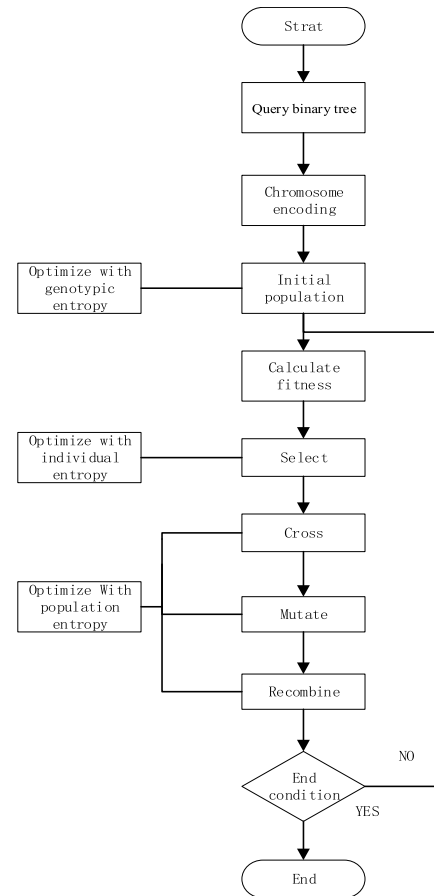


FIGURE 2. Flow chart of ADEGA.

query binary tree shown in Fig. 1 that contains relational tables R_1, R_2, R_3, R_4 and R_5 can be numbered as 1, 2, 3, 4, and 5, and the corresponding encoded string is 12345. Assume that the five tables are located at four different sites, and the corresponding sites from table 1 to table 5 are 1, 2, 2, 3, and 4. The site access order corresponding to the encoded string 12345 was 12234. When two adjacent access sites are the same, the join operation only needs to calculate the size of the intermediate data set after joining, and the transmission cost between sites is ignored. However, for different adjacent access sites, in addition to calculating the size of the intermediate data set, the transmission cost between sites is the main part of the query cost.

The goal of query optimization is to obtain the QEP with the lowest join cost, and the fitness value should show the advantages and disadvantages of the coding individuals corresponding to each join order. The smaller the join cost, the greater the corresponding fitness value. Therefore, the fitness value should be inversely proportional to the join cost. In this paper, we calculated the cost of each QEP according to equations (5) and (6), and the fitness function of each QEP can be obtained by taking the reciprocal of its cost:

$$f(x) = \frac{1}{COST(x)} \quad (7)$$

B. INITIAL POPULATION OPTIMIZATION

The initial population should be good at representing the entire solution space [25], and its distribution will directly affect the quality of the subsequent new population. In general genetic algorithms, the initial population is randomly generated. Due to randomness, the initial chromosome may be concentrated on a local area of the solution space, so it cannot represent the whole solution space, the population diversity is missed, and the genetic advantage is greatly reduced. To improve the diversity of the initial population, we used genotype entropy to assist in generating the initial population. Genotype entropy reflects the diversity of the individual loci. Assume that there is an initial population composed of M individuals with an encoding length of L (as shown in Fig. 3), x_j^i represents the j th gene of individual i in the population.

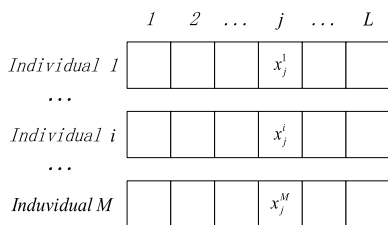


FIGURE 3. Diagram of group coding.

Genotype entropy H_j of the j th gene in the population can be defined as:

$$H_j = \sum_{k \in V_j} (-P_{jk} \log P_{jk})$$

$$P_{jk} = \frac{N_{jk}}{M} \tag{8}$$

In the formula, k represents the possible value of the j th gene, V_j is the set of k , which theoretically equals the encoding length L . Then, P_{jk} can be understood as the frequency at which the j th gene in the population is equal to the k value, that is, the ratio of the number N_{jk} of the k value to the population size M .

The genotype entropy H of the whole population is defined as the average value of all genotype entropies H_j in the population:

$$H = \frac{1}{L} \sum_{j=1}^L H_j \tag{9}$$

The specific generation process of the initial population is as follows:

- 1) In the individual definition domain, generate N_0 individuals randomly ($N_0 < N$) and calculate their entropy H_0 .
- 2) In the individual definition domain, generate an individual randomly, and then calculate the entropy H with the new individual and the existing individuals. If $H \geq H_0$, we will receive the new individual and update the value of H_0 to H ; otherwise, we will reject the new individual, regenerate another new individual randomly, and continue step 2) until $H \geq H_0$.

3) Repeat step 2) until the number of individuals in the initial population reaches the target number N .

For example, for the join query of five relational tables in Fig. 1, the query tree can be encoded into gene sequence 12345, which is one of the combinations in the solution space. Through the above steps, every turn can produce new gene sequence combinations that are different from all previous individuals in the solution space as much as possible. The retained individuals can ensure that the genotype entropy of the expanded population is not inferior to that of the previous population. The initial population generated in this way can be well distributed in the entire solution space, has good diversity, and can accelerate the speed of evolution.

C. GENETIC OPERATION

In the iterative process of the genetic algorithm, the fitness of the chromosome is the core of the judgment. In this paper, the fitness value was inversely proportional to the cost of the query execution plan. The higher the fitness value, the smaller the corresponding query cost, and the better the corresponding QEP. The fitness value is calculated by substituting the encoding sequence into the cost model and taking the reciprocal, which can be regarded as the phenotype of the chromosome. Therefore, this paper introduces the concept of phenotype entropy, referring to [26].

Definition 1 (Population Entropy): Population entropy represents the phenotype entropy of the entire population. Assume that S is the search space, the population of generation t is $P_t = \{x_t^1, x_t^2, \dots, x_t^N \in \hat{E}S^N$, N is the population size, and the subpopulation produced by the population of t generation is $O_t = \{x_t^{N+1}, x_t^{N+2}, \dots, x_t^{N+M}$. We define the active window W_t of generation t as: 1) the active window of the initial population is $w_0 = [l_0, u_0]$, l_0 is the lower limit of the fitness value of the initial population, and u_0 is the upper limit; 2) the active window of generation t is $w_t = [l_t, u_t]$, and the active window of generation $t + 1$ is $w_{t+1} = [l_{t+1}, u_{t+1}]$, where $l_{t+1} = \min(l_t, l_{ot})$, $u_{t+1} = \max(u_t, u_{ot})$, l_t and u_t are the lower and upper limits of the population fitness of generation t , l_{ot} and u_{ot} represent the lower and upper limits of the fitness value of the sub-population produced by generation t . Then, the active window is divided into K pieces equally. The range of the j th part in the population of t th generation can be expressed as:

$$[l_t + (j - 1) \cdot \frac{u_t - l_t}{K}, l_t + j \cdot \frac{u_t - l_t}{K}] \tag{10}$$

If the number of individuals falling into the j th interval of the active window w_t in the population P_t is n_j , then the individual density in this interval is $\frac{n_j}{N}$, the population entropy E of t th generation can be defined as:

$$E = - \sum_{j=1}^K p_j \cdot \log(p_j)$$

$$p_j = \frac{n_j}{N} \tag{11}$$

Definition 2 (Individual Entropy): In the population P_t of generation t , if the fitness value of chromosome i falls into the j th interval of the active window, the individual entropy ε_i of chromosome i can be defined as

$$\varepsilon_i = -\frac{1}{N} \log(p_{ij}) \quad (12)$$

p_{ij} is the individual density of the j th active interval in which chromosome i falls, and N is the population size.

Population entropy and individual entropy are interrelated. The additivity of entropy can be verified by equations (11) and (12), that is, the population entropy is equal to the sum of all individual entropies. Population entropy is a measure of population distribution in the macro, while individual entropy is the distribution of individuals in the micro.

1) SELECTION OPERATOR

In this paper, we combined the screening effect of fitness and individual entropy on the population, and the individual selection probability is formulated as follows:

$$p_i = \frac{f_i \cdot \exp(\varepsilon_i)}{\sum_{i=1}^N f_i \cdot \exp(\varepsilon_i)} \quad (13)$$

f_i is the fitness value of the i th individual in the population, ε_i is the individual entropy defined above. We used roulette [27] as the selection method to filter the population. When the population diversity is high, individuals with higher fitness values are more likely to be retained. While the population diversity is lost, the chance of individuals with small fitness values to be retained in the next generation increases. It can avoid the loss of effective genes to maintain population diversity.

2) CROSSOVER OPERATOR

In the genetic algorithm, the crossover operator is the main method for generating new individuals. In this paper, we consider the influence of evolutionary algebra and entropy on the crossover operator. In the early stages of evolution, a large crossover probability should be adopted to accelerate the generation of new individuals. However, in the later stage of evolution, the crossover probability should be reduced to prevent the destruction of the structure of excellent individuals. When the population entropy is large, the population's individual diversity is high. At this time, it is necessary to concentrate on mining the structure of better solutions, so the crossover probability should be increased. When the population entropy is small, the crossover probability should be reduced to avoid destroying the optimal solution structure. Therefore, the crossover probability P_C is set as:

$$P_c = P_{c0} \cdot \exp\left(-\frac{g \cdot H}{G \cdot E}\right) \quad (14)$$

P_{c0} is the initial crossover probability, we chose 0.9 in this paper, g is the current generation, G is the maximum evolution times, E is the population entropy of the current population, H is the maximum population entropy in theory.

For the two individuals x_1^t and x_2^t of the t generation, the value range of chromosome genes is $[a, b]$, and then the crossover operator generates sub-individuals x_1^{t+1} and x_2^{t+1} using the following method:

$$\begin{cases} x_1^{t+1} = 0.5[(1 + \beta)x_1^t + (1 - \beta)x_2^t] \\ x_2^{t+1} = 0.5[(1 - \beta)x_1^t + (1 + \beta)x_2^t] \end{cases} \quad (15)$$

where

$$\beta = \begin{cases} [(2 - \gamma^{-2})u]^{1/2}, & u \leq 1/(2 - \gamma^{-2}) \\ [2 - (2 - \gamma^{-2})u]^{-1/2}, & \text{others,} \end{cases}$$

$$\gamma = 1 + \frac{2}{x_2^t - x_1^t} \min[(x_1^t - a), (b - x_2^t)]$$

u is a random number uniformly distributed between $[0, 1]$.

3) MUTATION OPERATOR

When the algorithm iterates into a local area, it may fall into a local optimum. To make the algorithm jump out of the local area and continue to search globally, we adopted the following mutation strategy.

When the population entropy is large and the diversity of the population is high, the structure of the solution space is sufficient, and the mutation operator only needs to search for the optimal value in the current range. At this time, the probability and step length of the mutation should be reduced. Otherwise, when the entropy of the population is small, the diversity of the population is lacking, and it is necessary to increase the probability and step length of the mutation, so that the algorithm can jump out of the local area and search globally. As the number of iterations increases, the step length of the mutation should be reduced to prevent breaking the optimal solution structure that has been found, and the mutation probability should increase because the crossover probability is small in the later iteration, and the mutation operator will be used to assist the generation of new individuals. According to the above analysis, the mutation probability P_M is set as

$$P_M = P_{M0} \cdot \exp\left(\frac{g \cdot H}{G \cdot E}\right) \quad (16)$$

P_{M0} is the initial mutation probability, we chose 0.06 in this paper.

We use the following method to generate mutation individuals:

$$x' = \begin{cases} x + (u - x) \cdot r_1 \cdot \left(1 - \frac{E}{H}\right)^{0.1} \cdot \left(1 - \frac{g}{G}\right)^{0.1}, & r > 0.5 \\ x - (x - l) \cdot r_1 \cdot \left(1 - \frac{E}{H}\right)^{0.1} \cdot \left(1 - \frac{g}{G}\right)^{0.1}, & r \leq 0.5 \end{cases} \quad (17)$$

where x' is the mutation individual of x , r and r_1 are random numbers within $(0, 1)$, u and l are the upper and lower limits of the values of the chromosome gene.

4) REORGANIZATION OF FATHER AND SON POPULATIONS

To maintain a good diversity of the population during the evolution process, when the son and parent individuals are reorganized to form the next generation population, the change in population entropy will be used as the selection criteria to ensure that the population entropy of the next generation is not inferior to the previous generation in every turn. The specific steps are as follows.

1) Add all the parent individuals and all the son individuals generated after the genetic operation to form an intermediate population, calculate the fitness value of every individual, and sort by the fitness value from large to small.

2) According to the elite retention strategy, select the first N_0 individuals with the best fitness value from the intermediate population, record them as the temporary population P_0 , calculate its population entropy E_0 , and remove these N_0 individuals from the intermediate population.

3) Select the first individual from the remaining intermediate population and add it to P_0 , and calculate the population entropy E of the new P_0 . If $E \geq E_0$, retain the individual in P_0 and remove the individual from the intermediate population. Otherwise, the individual will not be retained, and select the next individual from the intermediate population and repeat step 3) until $E \geq E_0$.

4) Repeat step 3) until the number of individuals in the temporary population P_0 reaches the required N . Then, P_0 is the population of the next generation.

This method can keep the population diversity rich as much as possible while selecting excellent individuals for the next generation. This can increase the search speed of the algorithm.

V. SIMULATION RESULTS

In this section, we test the performance difference between the ADEGA algorithm and other comparison algorithms for distributed queries based on a distributed database composed of five servers. The dataset used in the experiment was a set containing more than 2 million records obtained from the Internet, which was divided into several tables and randomly allocated to the databases of five servers. All experiments were carried out on an Intel (R) 2.2GHz machine with 8G physical memory, all five servers using the CentOS 7 system and MySQL database.

The comparison algorithms selected in this paper were the adaptive genetic algorithm (AGA) [28] and the parallel ant colony algorithm (PACA) [29]. The parameters of the algorithms were set as follows: population size was 100, maximum iteration number was 500, initial crossover probability was 0.9, and initial mutation probability was 0.06. In the experiment, the communication performance among sites was represented by the following network performance matrix:

$$M = \begin{bmatrix} -1 & 1.3 & 2.5 & 1.7 & 3.1 \\ 1.3 & -1 & 1.8 & 0.9 & 2 \\ 2.5 & 1.8 & -1 & 1.1 & 3 \\ 1.7 & 0.9 & 1.1 & -1 & 1.4 \\ 3.1 & 2 & 3 & 1.4 & -1 \end{bmatrix}$$

In this experiment, we tested the distributed database query with 4, 6, 8, 10, and 12 tables.

First, to test the performance of the algorithm in this paper compared to the comparison algorithms, we used the three algorithms to process a join query of 10 tables. The **convergence diagram** for the three algorithms is shown in Fig. 4. When the curve becomes flat, it indicates that the algorithm has converged.

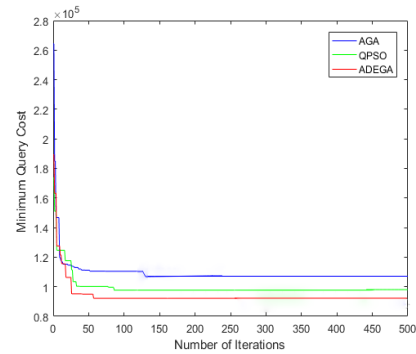


FIGURE 4. Convergence diagram of iteration.

It can be seen from the figure that when all three algorithms converged, the number of iterations when the ADEGA proposed in this paper starts to converge is the smallest, the number of iterations when PACA starts to converge is slightly larger than that of ADEGA, and the number of iterations when AGA starts to converge is the largest. It shows that compared with comparison algorithms, the algorithm proposed in this paper has a faster convergence speed and can search for the optimal solution more quickly. The ordinate in the figure represents the minimum cost in the population after each iteration. When all three algorithms converged, the minimum cost of the ADEGA proposed in this paper is the smallest, the minimum cost of PACA is slightly larger than that of ADEGA, and the minimum cost of AGA is the largest. It shows that the algorithm proposed in this paper is better than comparison algorithms, and its optimal solution that can be searched is closer to the global optimal solution. The above results show that the algorithm proposed in this paper has better convergence performance and results than comparison algorithms.

Second, for the distributed database query with the number of join tables of 4, 6, 8, 10, and 12, we performed the search using the three algorithms. We selected the amount of transmission data for the query execution plan to represent the query cost. Assume that when the minimum query cost does not change for 100 consecutive generations, the algorithm is considered to have converged. Record the minimum cost, iterations, and search time when each algorithm converges. The results are presented in Table 1, Table 2, and Table 3.

According to the results shown in Table 1, Table 2 and Table 3, we can conclude the following. When the number of join tables is 4, the optimization results of the three algorithms are almost the same. This is because when the number of join

TABLE 1. Comparison of minimum query cost (unit: kb).

Number of tables	4	6	8	10	12
GA	26.50	73.00	83.53	107.71	96.64
EGA	26.50	72.55	78.25	96.98	92.84
ADEGA	26.50	70.95	76.31	85.96	81.66

TABLE 2. Comparison of iteration number.

Number of tables	4	6	8	10	12
GA	105	117	167	178	216
EGA	104	110	139	155	186
ADEGA	102	108	122	145	178

TABLE 3. Comparison of search time (unit: s).

Number of tables	4	6	8	10	12
GA	5.62	11.44	19.61	30.59	42.37
EGA	5.55	9.61	15.32	24.27	28.50
ADEGA	5.38	7.89	12.90	18.67	20.52

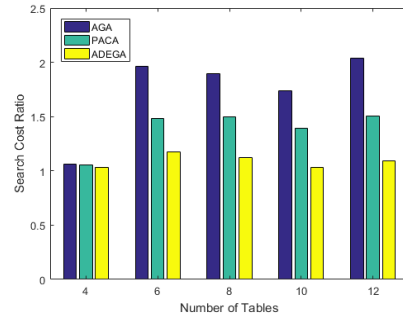
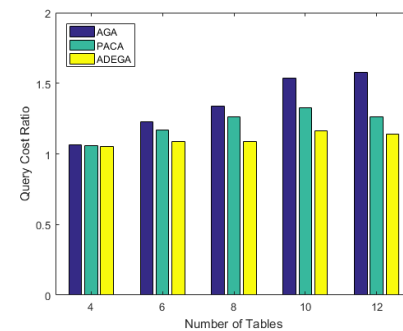
tables is 4, the number of different kinds of QEPs is only 24, which is smaller than the population size of 100. At this time, the initial population will contain almost all types of chromosome sequences, and all three algorithms can search for the global optimal QEP at the beginning. Thus, there is little difference among the three algorithms under this condition. However, as the number of join tables increases to 6, 8, and more, the search space of algorithms becomes increasingly large, which has already exceeded the population size and needs to search for the optimal solution gradually. Thus, the performance difference among the three algorithms became increasingly significant. Under these conditions, compared to AGA and PACA, ADEGA performed the best. The optimal solution searched by ADEGA has the lowest query cost, its number of iterations is the smallest, and its search time is also the smallest. PACA's results were the next, and AGA's were the worst. Moreover, the larger the number of join tables, the greater the gap between the ADEGA and the comparison algorithms. This indicates that the ADEGA proposed in this paper has the better optimization effect and query efficiency in distributed database multi-join queries.

Finally, we test the effect of the algorithm in this paper on distributed database queries. We applied the optimal query scheme found by the three algorithms in our distributed database environment, recorded the query time of each scheme, and used the following evaluation indicators:

$$\begin{aligned} & \text{search cost ratio} \\ &= \frac{\text{search time of current optimal query scheme}}{\text{search time of global optimal query scheme}} \quad (18) \end{aligned}$$

$$\begin{aligned} & \text{query cost ratio} \\ &= \frac{\text{execution time of current optimal query scheme}}{\text{execution time of global optimal query scheme}} \quad (19) \end{aligned}$$

The results were shown in Fig. 5 and Fig. 6.

**FIGURE 5.** Search cost ratios of all algorithms.**FIGURE 6.** Query cost ratios of all algorithms.

From Fig. 5 and Fig. 6, we can see that when the number of tables is 4, because the initial population may contain all different QEPs, the search cost ratios and query cost ratios of all three algorithms were very close to 1, and there was little difference in search performance among these three algorithms. However, as the number of join tables increases, compared to AGA and PACA, the ADEGA in this paper has a smaller search cost ratio and query cost ratio, which are closer to 1. This indicates that, as the number of join tables increases, the ADEGA can find the solution that is closest to the global optimal solution more quickly and efficiently, and the query efficiency is greatly improved. This is because the algorithm in this paper always maintains a good diversity of population in the iteration process, so that the algorithm can jump out of the local optimum and avoid the algorithm from falling into premature, to better search for the global optimal solution.

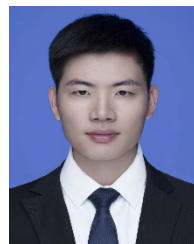
VI. CONCLUSION

Aiming at the premature problem that exists in multi-join queries in distributed databases using traditional genetic algorithms. In this paper, we propose an adaptive double-entropy genetic algorithm (ADEGA) based on genotype entropy and phenotype entropy. This algorithm optimizes the initial pop-

ulation distribution based on genotype entropy and adaptively selects genetic strategies based on phenotype entropy to maintain population diversity in the iteration process. The results of the experiment show that by maintaining the population diversity in the evolution process, this algorithm can be effectively prevented from falling into the local optimum, the global search ability is improved, and a better query execution plan can be obtained.

REFERENCES

- [1] M. T. Özsu and V. Patrick, *Principles of Distributed Database Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1999, pp. 144–154.
- [2] P. A. Bernstein, N. Goodman, E. Wong, C. L. Reeve, and J. B. Rothnie, Jr., “Query processing in a system for distributed databases (SDD-1),” *ACM Trans. Database Syst.*, vol. 6, no. 4, pp. 602–625, 1981.
- [3] B. J. Wang and S. Y. Wu, “The improved SDD-1 algorithm applied to Hive,” *Natural Sci. J. Xiangtan Univ.*, vol. 36, no. 4, pp. 77–82, 2014.
- [4] D. Kossmann and K. Stocker, “Iterative dynamic programming: A new class of query optimization algorithms,” *ACM Trans. Database Syst.*, vol. 25, no. 1, pp. 43–82, Mar. 2000.
- [5] P. Doshi and V. Raisinghani, “Review of dynamic query optimization strategies in distributed database,” in *Proc. 3rd Int. Conf. Electron. Comput. Technol.*, Apr. 2011, pp. 145–149.
- [6] A. K. Giri and R. Kumar, “Distributed query processing plan generation using iterative improvement and simulated annealing,” in *Proc. 3rd IEEE Int. Advance Comput. Conf. (IACC)*, Feb. 2013, pp. 757–762, doi: [10.1109/IAdCC.2013.6514322](https://doi.org/10.1109/IAdCC.2013.6514322).
- [7] L. Wang, R. Cai, M. Lin, and Y. Zhong, “Enhanced list-based simulated annealing algorithm for large-scale traveling salesman problem,” *IEEE Access*, vol. 7, pp. 144366–144380, 2019, doi: [10.1109/ACCESS.2019.2945570](https://doi.org/10.1109/ACCESS.2019.2945570).
- [8] T. V. V. Kumar, V. Singh, and A. K. Verma, “Generating distributed query processing plans using genetic algorithm,” in *Proc. Int. Conf. Data Storage Data Eng.*, Feb. 2010, pp. 173–177.
- [9] C. Lin, “An adaptive genetic algorithm based on population diversity strategy,” in *Proc. 3rd Int. Conf. Genetic Evol. Comput.*, Oct. 2009, pp. 93–96, doi: [10.1109/WGEC.2009.67](https://doi.org/10.1109/WGEC.2009.67).
- [10] Q. Liu, X. Liu, J. Wu, and Y. Li, “An improved NSGA-III algorithm using genetic K-means clustering algorithm,” *IEEE Access*, vol. 7, pp. 185239–185249, 2019, doi: [10.1109/ACCESS.2019.2960531](https://doi.org/10.1109/ACCESS.2019.2960531).
- [11] S. Deng, “Distributed database query optimization algorithm based on Tabu GEP,” *Comput. Digit. Eng.*, vol. 41, no. 10, pp. 1552–1555, 2013.
- [12] Y. Pan, “Application of adaptive genetic algorithm in query optimization of distributed database,” *J. Inner Mongolia Normal Univ. Natural Sci.*, vol. 45, no. 1, pp. 94–97, 2016.
- [13] Y. Zhou and H. J. Cheng, “Distributed database query optimization based on multi-ant colony genetic algorithm,” *J. Shanghai Normal Univ. Natural Sci.*, vol. 47, no. 1, pp. 37–42, 2018.
- [14] S. J. Dong and J. P. Wang, “Cheng Y. HMSST+: HMSST algorithm optimization based on distributed memory database,” *Comput. Sci.*, vol. 43, no. 3, pp. 220–224, 2016.
- [15] H. Li and B. Luo, “A tree-based genetic algorithm for distributed database,” in *Proc. IEEE Int. Conf. Autom. Logistics*, Sep. 2008, pp. 2614–2618, doi: [10.1109/ICAL.2008.4636613](https://doi.org/10.1109/ICAL.2008.4636613).
- [16] Z. H. Cheng, L. Hong, and J. Y. Qian, “Adaptive greedy GA algorithm for TSP,” *Comput. Sci.*, vol. 39, no. 6, pp. 184–187, 2012.
- [17] H. Yu and X. K. Wang, “Distributed query optimization algorithm based on value,” *J. Dalian Univ. Technol.*, vol. 3, pp. 453–458, Jan. 2005.
- [18] X. B. Shuai, S. N. Ma, X. G. Zhou, and A. Gong, “Research on query optimization of distributed database based on genetic algorithm,” *Small Microcomput. Syst.*, vol. 30, no. 8, pp. 1600–1604, 2009.
- [19] L. B. Booker and R. Riolo, “Introduction to the special issue: Advances in genetic algorithms—Research trends and perspectives,” *Evol. Comput.*, vol. 8, no. 4, pp. 3–4, Dec. 2000, doi: [10.1162/106365600568211](https://doi.org/10.1162/106365600568211).
- [20] F. Liu, Y. L. Ma, and H. J. Zhou, “Adaptive genetic algorithm optimization simulation based on population diversity,” *Comput. Simul.*, vol. 34, no. 4, pp. 250–255, 2017.
- [21] Y. X. Shen and C. F. Zhang, “A novel genetic algorithm for preserving population diversity,” *J. Syst. Simul.*, vol. 17, no. 5, pp. 1052–1053, 2005.
- [22] R. Takahashi, “Verification of thermo-dynamical genetic algorithm to solve the function optimization problem through diversity measurement—Diversity measurement and its application to selection strategies in genetic algorithms,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 168–177, doi: [10.1109/CEC.2016.7743792](https://doi.org/10.1109/CEC.2016.7743792).
- [23] L. Jiacheng and L. Lei, “A hybrid genetic algorithm based on information entropy and game theory,” *IEEE Access*, vol. 8, pp. 36602–36611, 2020.
- [24] Z. Shen, Y. Hao, and K. Li, “Application research of an adaptive genetic algorithms based on information entropy in path planning,” in *Proc. IEEE Int. Conf. Inf. Autom.*, Jun. 2010, pp. 2013–2016, doi: [10.1109/ICINFA.2010.5512030](https://doi.org/10.1109/ICINFA.2010.5512030).
- [25] Y. Zhu and C. P. Han, “A fast genetic algorithm for adaptive convergence of population,” *Comput. Sci.*, vol. 39, no. 10, pp. 214–217, 2012.
- [26] W. Q. Ying and Y. X. Li, “Improvement of computational efficiency of thermodynamic genetic algorithm,” *J. Softw.*, vol. 19, no. 7, pp. 1613–1622, 2008.
- [27] A. Lipowski and D. Lipowska, “Roulette-wheel selection via stochastic acceptance,” *Phys. A, Stat. Mech. Appl.*, vol. 391, no. 6, pp. 2193–2196, Mar. 2012.
- [28] C. Yang, Q. Qian, F. Wang, and M. Sun, “An improved adaptive genetic algorithm for function optimization,” in *Proc. IEEE Int. Conf. Inf. Autom. (ICIA)*, Aug. 2016, pp. 675–680, doi: [10.1109/ICInfA.2016.7831905](https://doi.org/10.1109/ICInfA.2016.7831905).
- [29] W. Zheng, X. Jin, F. Deng, S. Mo, Y. Qu, Y. Yang, X. Li, S. Long, C. Zheng, J. Liu, and Z. Xie, “Database query optimization based on parallel ant colony algorithm,” in *Proc. IEEE 3rd Int. Conf. Image, Vis. Comput. (ICIVC)*, Jun. 2018, pp. 653–656.



BINGXU ZHENG was born in Taizhou, Zhejiang, China, in 1996. He received the B.S. degree in information science and technology from Shanxi University, Taiyuan, China. He is currently pursuing the M.S. degree with the Information Engineering College, Zhejiang University of Technology. His research interests include distributed database and multimedia communication.



XIANG LI was born in 1973. He received the M.S. degree. He is currently working as a Senior Engineer at Zhongtong Fuhuizhan Technology Company, Hangzhou. His main research interests include exhibition application software, data communication, and the Internet of Things.



ZHENZHEN TIAN was born in Tai'an, Shandong, China, in 1995. She received the B.S. degree in electronic information science and technology from the Texas College, Shandong. She is currently pursuing the M.S. degree with the Information Engineering College, Zhejiang University of Technology. Her main research interest includes multimedia communication.



LIMIN MENG received the Ph.D. degree in communication and electronic systems from Zhejiang University, Zhejiang, China. She is currently a Professor with the School of Information Engineering, Zhejiang University of Technology. Her main research interests include wireless communication and networks, streaming media transmission, and the IoT communications. She is a member of the China Computer Federation (CCF).