SQL Optimization in Oracle using Hybrid Genetic and Ant Colony Algorithm

Rajan Kusi¹, Kabir Kumar Sinkemana², Sanjeeb Prasad Panday¹* and Shashidhar Ram Joshi¹

¹Department of Electronics and Computer, Institute of Engineering Pulchowk Campus, Lalitpur
²Nepal College of Information Technology, Balkumari, Lalitpur

ABSTRACT

In this paper, the input user Structured Query Language (SQL) query is converted into an optimized SQL query using a hybrid algorithm. The main aim is to reduce query execution time using PHP language and oracle database. These performance has been evaluated using different performance metrics: Cost of individuals, Query execution time. The hybrid algorithm method combines the evolutionary effect of the Genetic Algorithm (GA) and the cooperative effect of Ant Colony Optimization (ACO). A GA with a great global converging rate aims to produce an initial optimum for allocating initial pheromones of ACO. An ACO with great parallelism and effective feedback is then served to obtain the optimal solution. A fused algorithm of a GA and ACO to solve SQL optimization problems is an innovative solution that presents a clear methodological contribution to the optimization algorithm. In the simulation result, we found the algorithm of a GA and ACO to solve SQL optimization problems in Oracle. It is an innovative solution that presents a clear methodological contribution to the optimization algorithm.

Keywords: Ant Colony, Genetic, SQL, Query Optimization, Query Execution plan
1. INTRODUCTION
Structured Query Language (SQL) Statements are the specialized language used to retrieve data from the database. We can get the same results by writing different SQL queries. However, using the best query is important when performance is considered. Making a query run faster reduces the number of calculations that the software (and, therefore, hardware) must perform. We need some understanding of how SQL makes calculations. First, let us address high-level things that will affect the number of calculations needed to make and, therefore, query run time: table size, joins, and aggregations.

Genetic algorithms (GAs) are adaptive metaheuristic search algorithms classified as evolutionary computing algorithms. It is a technique inspired by natural evolution that Charles Darwin postulated as a theory of natural evolution (Yan 2021; Holland 1975). This algorithm reflects the process of natural selection, where the fittest individuals are selected for reproduction to produce offspring of the next generation. The phases considered in a genetic algorithm are Initial population, Fitness function, Selection, Crossover, and Mutation.

Ant colony algorithms are becoming popular approaches for solving combinatorial optimization problems in database SQL systems. The basic idea of ant heuristics is the natural behavior of ants that succeed in finding the shortest paths from their nest to food sources. They communicate via a collective memory consisting of pheromone trails (Wagh & Nemade 2017; Chande & Sinha 2010; Pan & Wang 2009). As ants have weak global insight into their environment, they move randomly when no pheromone is available. However, it tends to follow a path with a high pheromone level when many ants move in a common area, leading to an autocatalytic process. An ant does not choose its direction based on the level of pheromone exclusively but also considers the neighborhood of the nest and the food place, respectively, into account. It allows the discovery of new and potentially shorter paths.

2. MATERIALS AND METHODS

2.1 Hybridization Algorithm of a GA and ACO
A system has been implemented combining the Ant Colony algorithm and Genetic algorithm. GA has strong flexibility and quick global searching ability with a higher population scattering ability for the extensive amplitude of answers. On the other hand, ACO has a low population scattering ability, parallel processing, high convergence speed, and global searching ability with a positive feedback mechanism. By combining both algorithms, strong features of both can improve the performance while overcoming the drawbacks of each. The performance of GA depends on the population and operator used in the algorithm. If the population size is small, then it converges fast.

2.2 Basic Principle of Adynamic Combination of GA and ACO
“Fig 1” shows the speed-time curve of the genetic algorithm and ant colony algorithm. For the genetic algorithm, there is a higher convergence speed to the optimal solution during the preliminary stage (t0–ta), but it will significantly reduce after ta. However, during the preliminary stage (t0–ta) of the ant colony algorithm, the searching speed is very slow for lacking pheromones. Then after pheromones reach a certain degree (after ta), the convergence speed to optimal solution improves quickly. The basic principle of a dynamic combination between a genetic algorithm and an ant colony algorithm is that we can utilize a genetic algorithm to get initial pheromones. After that, obtain the optimal solution by ant colony algorithm (Zhao et al. 2016).

Fig. 1. Difference in the speed-time curve of GA and ACO
2.3 Hybridization of GA and ACO

Fig. 2. Hybrid GA and ACO

Fig. 2 shows the block diagram of the hybrid GA and ACO process that consists of various steps. Query translator converts the high-level input language (SQL) into relational algebra expression. Then it is converted into query tree structures. Different query trees can be generated from a single input query. These query trees are treated as query execution plans (QEPs). The solution of a query is the set of all QEPs for the query.

(Butey et al. 2012; Horng et al. 1994; Petković 2010) In GA, the initial population is randomly selected and will contain certain chromosomes. Each chromosome is again validated using the fitness function, and each individual’s fitness is calculated. Evaluation of fitness is applied to eliminate the worst-fitted members in the population. Once the initial population is created, two fit chromosomes are selected from a population that can mate and create their offspring. The fit chromosomes are selected, allowing them to produce offspring in such a way that would lead to chromosomes closer to the optimal solution and, therefore, less diversity. The primary transformation is performed by crossover and mutation. One point crossover is used, a random crossover point is selected, and the tails of both chromosomes are swapped to produce new offspring. The crossover probability and mutation probability are calculated for this operation. The remaining members in the population are given to the ant colony optimizer. Here, the node of the query tree represents a state, and the tree’s edge represents a path length. The initial pheromones for the routes of ACO are settled according to the results of the GA. When artificial ants move from one state to another, the path length between these two states is calculated. If a single ant travels all possible states of the query, then the tour is completed. After completing a tour of the artificial ant, a pheromone is updated for each ant. Once the stop criteria are met, the output result will be the best solution for the input SQL query.

Query translator converts the high-level input language (SQL) into relational algebra expression (Ceri & Gottlob 1985). Then it is converted into query tree structures. Different query trees can be generated from a single input query. These query trees are treated as query execution plans (QEPs). The solution of a query is the set of all QEPs for the query. Each execution plan can be considered a solution program for the problem of finding a good access path to retrieve the required data.
1. Estimated cost model for fitness function
The simple cost model is based on two assumptions (Dong & Liang 2007)
a) The uniform distribution of attribute values and
b) The sum of the size of the intermediate relations determines the cost of a join tree.
The cost of the join tree is

$$\text{Cost} = \sum_{i=0}^{n-1} n(t_i)$$  \hspace{1cm} (1)

The Number of tuples is given by

$$n(t) = \frac{n(r) \times n(s)}{\pi C_j \in C \max \left( V(C_j, r), V(C_j, s) \right)}$$  \hspace{1cm} (2)

And the Number of distinct values that appear in the relation r for attribute A

$$V(A, t) = \begin{cases} V(A, r) & A \in r - s \\ V(A, s) & A \in s - r \\ \min (V(A, r), V(A, s)) & A \in r \ A \in s \end{cases}$$  \hspace{1cm} (3)

2. Crossover and Mutation Probability.
The self-adaptive crossover and mutation probability functions are as follows:

$$P_c = \begin{cases} P_{c0} & , f \leq \bar{f} \\ P_{c1} \left( P_{c0} \left( \frac{f_{max} - f}{f_{max} - \bar{f}} \right) \right) & , f > \bar{f} \end{cases}$$  \hspace{1cm} (4)

$$P_m = \begin{cases} P_{m0} & , f' \leq \bar{f} \\ P_{m1} \left( P_{m0} \left( \frac{f_{max} - f'}{f_{max} - \bar{f}} \right) \right) & , f' > \bar{f} \end{cases}$$  \hspace{1cm} (5)
2.4 Ant Colony Algorithm Technique Used in Query Optimization

Fig. 3. First step of the tour of the ant colony algorithm

Fig. 4. Second step of the tour of the ant colony algorithm

Fig. 5. Third step of the tour of the ant colony algorithm
Fig. 3, 4, and 5 show the various steps of the tour of the ant colony algorithm. The four join tables, i.e., A, B, C, and D, denote state, and their relationship as the path, are considered. Artificial ants start their tour from table A. After reaching the next state, the estimated cost and execution time are calculated. Here joining tables A-C has a minimum estimated execution time than joining tables A-B and A-D. Then different ants continue the tour from combined table A-C to the remaining state i.e., table B and table D, and calculate the estimated cost and execution time. The join tables A-C-D has minimum execution time compared to join tables A-C-B.

Now only one node i.e., table B, remains. The tour is completed when the ant reaches node B. Finally, the estimated cost and execution time is calculated for the query with these four join tables. This way, an optimized query is obtained after artificial ants complete all possible tours.

3. SIMULATION AND RESULT

The input parameters considered in simulation and experiments are the input SQL query, the Number of the initial population, the initial population, the number of selected populations, the Number of maximum generations, and the cut-off time.

Fig. 6. Query optimization processing form

Fig. 6 is a form designed for the query optimization process. For the query optimization process, five input parameters are considered. These are; input SQL query, number of the initial population, number of selected population, number of maximum generation, and cut-off time.
The result of a query with four join tables for original input query, ACA, GA, and Hybrid ACA and GA for conditioning experiment on 5 million tuples records are obtained as below:

<table>
<thead>
<tr>
<th>No. of join tables</th>
<th>Original (Input)</th>
<th>ACA</th>
<th>GA</th>
<th>Hybrid GA and ACA</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>56 sec</td>
<td>53 sec</td>
<td>22 sec</td>
<td>19 sec</td>
</tr>
<tr>
<td>5</td>
<td>57 sec</td>
<td>54 sec</td>
<td>22 sec</td>
<td>19 sec</td>
</tr>
<tr>
<td>6</td>
<td>57 sec</td>
<td>54 sec</td>
<td>22 sec</td>
<td>19 sec</td>
</tr>
<tr>
<td>7</td>
<td>59 sec</td>
<td>56 sec</td>
<td>24 sec</td>
<td>19 sec</td>
</tr>
<tr>
<td>8</td>
<td>61 sec</td>
<td>59 sec</td>
<td>25 sec</td>
<td>20 sec</td>
</tr>
<tr>
<td>9</td>
<td>66 sec</td>
<td>63 sec</td>
<td>35 sec</td>
<td>21 sec</td>
</tr>
<tr>
<td>10</td>
<td>68 sec</td>
<td>64 sec</td>
<td>35 sec</td>
<td>21 sec</td>
</tr>
<tr>
<td>11</td>
<td>70 sec</td>
<td>66 sec</td>
<td>35 sec</td>
<td>24 sec</td>
</tr>
<tr>
<td>12</td>
<td>70 sec</td>
<td>67 sec</td>
<td>35 sec</td>
<td>24 sec</td>
</tr>
<tr>
<td>13</td>
<td>80 sec</td>
<td>67 sec</td>
<td>35 sec</td>
<td>24 sec</td>
</tr>
<tr>
<td>14</td>
<td>86 sec</td>
<td>67 sec</td>
<td>35 sec</td>
<td>25 sec</td>
</tr>
<tr>
<td>15</td>
<td>90 sec</td>
<td>76 sec</td>
<td>43 sec</td>
<td>25 sec</td>
</tr>
<tr>
<td>16</td>
<td>90 sec</td>
<td>76 sec</td>
<td>43 sec</td>
<td>28 sec</td>
</tr>
<tr>
<td>17</td>
<td>92 sec</td>
<td>78 sec</td>
<td>43 sec</td>
<td>28 sec</td>
</tr>
<tr>
<td>18</td>
<td>98 sec</td>
<td>81 sec</td>
<td>43 sec</td>
<td>28 sec</td>
</tr>
<tr>
<td>19</td>
<td>99 sec</td>
<td>82 sec</td>
<td>45 sec</td>
<td>32 sec</td>
</tr>
<tr>
<td>20</td>
<td>110 sec</td>
<td>90 sec</td>
<td>45 sec</td>
<td>32 sec</td>
</tr>
</tbody>
</table>

Table 1. Comparison of the execution time of GA, ACA, and Hybrid algorithm

Fig. 7 shows the graphical representation of execution time vs. number of join tables for original input query, ACA, GA, and Hybrid algorithm.
4. CONCLUSION

This paper mainly focuses on the generation of population, then selecting the best population with the least execution time for parent query and performing crossover of the best population in Oracle. It has been found that the execution time for an optimized query using a hybrid genetic algorithm and ant colony algorithm in Oracle is less than the original query. Also, the next performance parameter, i.e., the CPU cycle for an optimized query using hybrid GA and ACA, is less than that of the original query. The SQL query optimization in the oracle database using a hybrid genetic algorithm and ant colony algorithm has been performed.

ACKNOWLEDGMENT

The authors are highly thankful to the Nepal College of Information Technology (NCIT) and the Institute of Engineering, Tribhuvan University, for providing us with this opportunity and guidance. The authors also thank the referees for their careful reading and valuable suggestions.

REFERENCES


Holland, J. H., 1975. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, 55 Hayward St., Cambridge, MA, USA.


Petković, D. š. 2010. Comparison of Different Solutions for Solving the Optimization Problem of Large Join Queries. IEEE. https://doi.org/10.1109/DBKDA.2010.1


