International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V70I8P216
Volume 70 Issue 8, 158-165, August 2022

Original Article

Prime Learning – Ant Colony Optimization Technique for Query Optimization in Distributed Database System

Praveena Mydolalu Veerappa[1], Ajeet Annarao Chikkamannur[2]

[1]*Department of Computer Science and Engineering, Dr Ambedkar Institute of Technology, Karnataka, India*
[2]*Department of Computer Science and Engineering, Nagarjuna College of Engineering and Technology, Karnataka, India*

[1] mv_praveena@rediffmail.com

Received: 10 June 2022 Revised: 26 July 2022 Accepted: 01 August 2022 Published: 22 August 2022

Abstract - *Query optimization is an important factor in a distributed database system that finds the parameters for the optimal execution of a plan to reduce the runtime. Query optimization is a challenging task in a distributed system that helps improve the database's efficiency. Various pieces of research applied Ant Colony Optimization (ACO) based method to improve query optimisation performance. The Prime Learning - Ant Colony Optimization (PL-ACO) method is applied to increase query optimisation performance in this research. The Prime strategy method is applied in the ACO method to replace the worst learner with the best learner to improve the learning rate. The key idea of the proposed method is to use the Prime strategy on Ant Colony Optimization to reduce the search space-time for query joins in a distributed database system. For Query access, processing and resource processing cost, the fitness function is created for each possible query join solution. For fitter functions, queries are being taken for processing in the next round, and weaker ants are eliminated. This process reduces the number of ants at each iteration, and the optimal solution is achieved quicker. Various queries in the database were used to test the efficiency of PL-ACO in query optimization. The system model is developed to apply the query in the system and evaluate the cost of the model. The proposed PL-ACO method has 10 iterations for 0.2 cost and existing ACO method has 68 iterations, and the ACO-Genetic Algorithm (GA) method has 65 iterations.*

Keywords - *Genetic Algorithm, Learning Rate, Prime Strategy, Query optimization.*

1. Introduction

Multi-Query Optimization (MQO) is finding the essential keyword in a database system for query processing to find the related information. MQO method explains answers to a set of queries with a common task, and each query has alternate executive plans. Each plan has a set of common tasks among several queries [1]. The cost-based query optimization uses cost as a factor to provide answers to the queries. The query uses various methods based on sorting, conditions and indexes. Integrating tables with join orders cost was used for the query optimizer computed only once. Table integration of least cost is used in the database and for query optimization in the future process [2]. The straightforward way to break through query limitation is to reduce privacy cost in each query, and more query is answered in privacy protection. The privacy cost minimization and privately answer a query with the urgent issue in data publishing [3]. The MQO method finds similarities among a set of queries, and various methods were used to avoid the search process in query execution. The MQO method provides small optimization overheads to increase query performance using pipelining, materialized views selection and sharing sub-expressions [4]. Low-level expressions from high-level language are mapped in the query in major impulsion of query processing. The low-level expressions are set to find a strategy for query execution [5].

The database community has studied the MQO, a well-known database research problem. The MQO method constructs a global plan to take advantage of common tasks, and many algorithms are proposed, such as improved and heuristic algorithms, to solve this problem [6, 7]. Different forms can be used to access a database based on a query, and query clauses of order changing do not change query results that can change response time. In querying a large database, response time is important. The query optimization method involves finding a query path to reduce the joined number and evaluating a query to respond as soon as possible to improve query run time [8 – 10]. Existing query optimization techniques have limitations of lower learning and local optima trap that degrades the performance. The objectives and contribution of the research are discussed below:

1. Prime strategy is applied to the Ant Colony Optimization method to replace the worst learner with the best learner, which helps improve the

model's learning efficiency. The Prime strategy with the ACO method reduces the search space-time for query joins in the distributed database system.
2. A fitness function is created for each possible query join solution that helps Prime strategy with ACO to find the solution with less time and reduces resource processing costs in the system.
3. Fitter functions measured in the optimization method are considered for processing in the next round, and weaker ants are removed. The number of ants is removed at each iteration, achieving an optimal solution quicker.

This paper is organized as follows: Query optimization of the literature review is provided in Section 2, and the proposed method is explained in Section 3. The simulation setup is provided in Section 4, and the result is provided in Section 5. The conclusion of this research paper is provided in Section 6.

2. Literature Review

The query optimization is important for a database management system that performs the appropriate process for user query execution. Some of the research involved in the optimization of join queries were discussed in this section.

Panahi and Navimipour [11] proposed a genetic operator and artificial bee colony to increase efficiency for join query optimization in database management. The global-local search capability and genetic operator were combined to create a candidate solution to improve Artificial Bee Colony (ABC) algorithm performance. The query cost evaluation was reduced, and the Top-K query quality plan was improved. The genetic operator increases the variety of solutions and reduces a query plan creating cost. The developed method reduces the optimisation overhead of the current query plan method. The developed method has the limitation of a trap into local optima and poor convergence, providing priority to local search. The Artificial Bee Colony method processing cost is more due to increases in the number of local features in the search.

Jafarineja and Amini [12] proposed an enhanced ACO algorithm to reduce the required processing efforts in multi-join query optimization. A bucked-based encrypted database generated total false-positive results in the proposed model. The enhanced method provides a quality solution with less response time. The developed method significantly reduces the multi-join query processing effort and the total amount of false-positive results. The method prevents pointless operations and produces feedback to find the best solution from useful meta-data. The developed method finds a better solution faster than successful methods in query optimization. The developed method was not applicable for large datasets due to the method solution rearranging the bucket around a certain level.

Mohsin [13] proposed a quantum-inspired ACO method based on the probabilistic algorithm of the hybrid method in distributed database model to increase the query join cost. The best trial selection in a large search space for diversified ability in quantum computing. The developed model was suitable for the distributed database and has higher efficiency in query optimization. The result of the developed method shows that the developed method handles simple and complex queries with less cost than the existing method. The searching efficiency of the model was lower than the existing method in query optimization. The quantum-inspired model applies a higher number of candidate solutions for the search process, which tends to increase the cost of processing.

Gao [25] proposed Histogram and Tuple Size in Join Multi-query Optimization (J-MOTH) for sharing data granularity, join granularity and implicit sort. The sort-based optimization was applied to implicit sort among join queries, and an additional model was used to optimize shared work. The developed method avoided the implicit sort operation and joined the redundant result. The result shows that the developed J-MOTH performs better than the naïve and fine-grained technique. The developed method has the limitation of a trap into local optima and poor convergence, providing more priority to exploration.

Kumar and Jha [15] proposed the ACO-GA method and Hadoop Distributed File System (HDFS) map-reduce method to improve query optimization in Big Data. The developed method consists of two phases: Big Data (BD) arrangement and query optimization phase. The SHA-512 algorithm was applied to pre-process the input data finding the Hash Value and HDFC map removal function to remove repeated data. The features such as confidence, support and frequent pattern are extracted. The entropy calculation was used to manage confidence and support. In the second phase, BD queries are collected, and the same features are extracted. The model losses the potential solution due to less efficiency in the exploitation process in the optimization model.

Mohsin [16] applied Quantum Inspired Ant Colony Algorithm (QIACO), a probabilistic algorithm of a hybrid method that was devised to improve joins query cost in the distributed dataset. Quantum computing can expand and diversify to cover large query search spaces. This process helps avoid falling into a local optimum, speed up the convergence, and select the best trails. The method identifies the join order to reduce total execution time based on faster convergence of Quantum inspired ant colony model. The method's search space is high, affecting the model's query cost.

Zheng [17] developed a mathematical model for query optimization in a distributed database and applied an Adaptive Genetic Algorithm On Double Entropy (ADEGA). Two types of entropy, such as phenotype and genotype, are applied to a genetic algorithm. The genotype entropy method is applied for initial population distribution, and good population diversity ensures the initial population. The genetic strategy is applied to optimize using Phenotype entropy that divides initial and population entropy. The double entropy method increases the query cost of the model.

3. Proposed Method

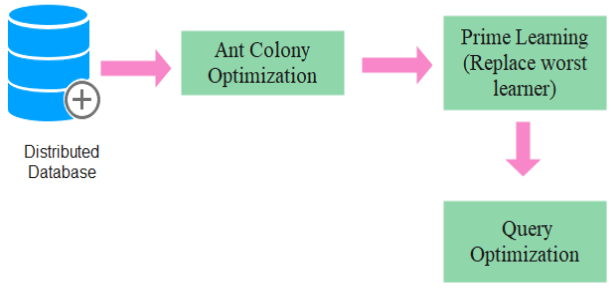

Fig. 1 The block diagram of the proposed PL-ACO method in query optimization

The system model is created in the proposed method to optimize the query and test the cost value of the method. This research proposes the Prime Learning Ant Colony Optimization (PL-ACO) method for query optimization. The Prime strategy is applied in the PL-ACO method to replace the worst learner with the best learner to improve the model learning rate. The PL-ACO in query optimization is shown in Fig. 1.

3.1. Ant Colony Optimization with Prime strategy

The number of iterations is set for the ACO algorithm. Many ants collected experiences of previous populations, and heuristic information was used to construct complete solutions in each iteration. The pheromone trail consists of elements of a solution to represent collected experiences. The components are deposited with pheromone, and the solving problem is based on connection [18, 19].

3.1.1. The Transition Rule

Ant acts as a computational agent in the ACO method and develops a solution for the problem at hand [20 – 24]. At each iteration of the algorithm, each ant moves from r to $s$state related to the complete intermediate solution. The unvisited states memorized in $J_r^k$the kth and from state r to state $s$is based on Eq. (1).

$$s = \underset{u \in J_r^k}{argmax}\left[\tau_i(r,u)^{\alpha}.\eta(r,u)^{\beta}\right], if q \le q_0(Exploitation) \quad (1)$$

Move desirability of posterior indication denotes trail level. Trails are updated in ants to complete the solution, trails level of increasing or decreasing is related to moves that were part of 'bad' or 'good' solutions.

Generally, the probability $p_k(r,s)$ of the kth ant moves from $rtos$state, as in Eq. (2).

$$p_k(r,s) = \begin{cases} \frac{\tau(r,s)^{\alpha},\eta(r,s)^{\beta}}{\sum_{u \in J_r^k}\tau(r,u)^{\alpha},(r,u)^{\beta}} & if s \in J_r^k \\ 0 & otherwise \end{cases} \quad (2)$$

The uniform probability is denoted as $q$is in range of [0, 1], the control parameters are denoted as $\alpha and \beta$the kth ant in the ith population of the set of the unvisited state is denoted asJ_r^k, the trail length from the state $r$and state u is denoted as $\eta(r,u)$, ith iteration pheromone concentration between state r and the state $u$is represented as $\tau(r,u)$, and the transition probability is denoted as $p_k(r,s)$.

3.1.2. The Pheromone Update Rule

The pheromone trails are measured to improve solution quality. The local and global updating trails are updated. The formula updating local trail is described as follows in Eq. (3).

$$\tau(r,u) = (1-\rho)\tau(r,s) + \sum_{k=1}^{m}\Delta\tau_k(r,s) \quad (3)$$

The evaporation rate of the pheromone trail is denoted as $\rho(0 < \rho < 1)$. The pheromone trail amount is denoted as $\Delta\tau_k(r,s)$ and ant k between time$t$and $t+\Delta t$in the tour is added to the edge (r,s) , as given in Eq. (4).

$$\Delta\tau k(r,s) = \begin{cases} \frac{Q}{L_k} & (r,s) \in \pi_k \\ 0 & otherwise \end{cases} \quad (4)$$

The constant parameter is denoted where ant toured with sequence and distance Q.

3.2. Prime Strategy

This research developed an ACO with a Prime strategy to improve this study's basic ACO search capability. The proposed method is simpler, easier to implement, and costs less than the basic ACO method. No need to manually tune parameters in PL-ACO like basic ACO except for iteration and population size. The teacher phase is simplified and redefined in the proposed PL-ACO, while the learner phase remains unchanged. The Prime strategy is applied to update the current worst learner, and local search operation is improved. The search phase is expressed in Eq. (5).

$$New(1,k) = \begin{cases} \tau_{ef}(k,k+1) & if rand < \mu \\ teacher & else \end{cases} \quad (5)$$

Where $rand$ is a uniformly distributed real value in a range of [0, 1], the mutation probability is denoted as m, the local search operation in random mutation variable is denoted asmu, the solution kth decision variable in solution is denoted as $kandk \in \{1,2,\ldots,Dim\}$, the current best teacher is as a new solution which is denoted asNew.

A better solution around the current best is searched using a local search attempt through a mutation variable. The Prime technique selects a new solution and replaces the worst solution, formulated as given in Eq. (6).

$$Wst = \begin{cases} New, & iff(New) < f(Wst) \\ Wst & else \end{cases} \quad (6)$$

The corresponding solution of the objective function is denoted as $f(.)$, and a current population of the worst one is denoted as Wst.

The elitist strategy is applied in ACO, and the solution quantity update is given in Eq. (7) and (8).

$$\tau_{ef}(k, k+1)\rho \times \tau_{ef}(k) + \Delta\tau_{ef}(k, k+1) + \tau_{ef}^{*} \quad (7)$$

$$\Delta\tau_{ef}^{*} = \sigma . g_f^{*} Q \quad (8)$$

The pheromone amount increases on the path $(e, f)$are represented $\Delta\tau_{ef}^{*}$ in Eq. (7) and (8) based on Prime ants. The Prime strategy with task assigning benefit value is denoted as $g_f^{*}$the number of Prime ants is denoted asσ.

4. Simulation Setup

The implementation details of the proposed method, such as dataset, and system configurations, are given in this section.

System Configuration: The implementation is carried out in the system consisting of windows 10 OS, Intel i9 processor, graphics card of 22 GB, and RAM of 128 GB. The Python 3.7 tool is used to evaluate the efficiency of the proposed method.

5. Results

The performance of the PL-ACO method in query optimization is calculated and compared with existing methods. The number of iterations for various costs and plans is measured and compared with existing query optimization methods.

Table 1 shows the working scenario, allocation of four scenarios, and logical sites of the dataset. Scenario R1 has a relation with various sites such as S1, S2, S5 and S6; scenario R2 has a relation with various sites such as S1, S2, S7, and S8.

Table 1. Query Cost

QP	QAC	QPC	RPC	COST= (QAC+QPC+RPC) /3
{6,8}	0.3761	0.1672	0.112	0.219
{6,5}	0.3764	0.2238	0.1122	0.2379
{8,5}	0.3767	0.279	0.112	0.2551
{3,5}	0.3762	0.335	0.1129	0.274
{8,8}	0.5017	0.224	0.1125	0.2789
{3,3}	0.5018	0.3348	0.1127	0.3153
{6,4,8}	0.5636	0.223	0.1675	0.3185
{6,7,8}	0.5644	0.2781	0.167	0.3368
{6,8,5}	0.5636	0.3343	0.1674	0.3552
{7,8,5}	0.5638	0.3899	0.168	0.3737
{6,8,8}	0.6891	0.3892	0.1671	0.379
{7,3,5}	0.5645	0.2787	0.167	0.3922
{7,8,8}	0.6892	0.4458	0.1673	0.397
{5,1,3}	0.564	0.5014	0.168	0.4109
{7,7,5}	0.6894	0.39	0.167	0.415
{7,3,3}	0.6895	0.4453	0.1675	0.4337
{6,7,4,8}	0.7519	0.3347	0.2232	0.4365
{5,3,5}	0.6894	0.5014	0.1673	0.453
{6,7,4,5}	0.752	0.389	0.2232	0.4548
{6,7,8,5}	0.7511	0.4455	0.2233	0.4735

The cost of queries is shown in Table 1 for various queries. The individual cost of the query and the total cost of the query are also shown in Table 2. This shows that the two sets of queries have a lower cost than the three.

Table 2. Number of iterations taken to attain each level of cost

Cost	ACO	ACO-GA	GA	PL-ACO
0.2	73	67	68	13
0.3	68	64	59	9
0.4	63	60	42	7
0.5	60	49	40	6
0.6	57	42	35	7
0.7	55	38	22	5

Fig. 2 and Table 2 show that the proposed and existing method requires some iteration to attain the query cost level. The number of iterations for various cost values is measured and compared with existing methods. This shows that the number of iterations is less for higher cost value in the query optimization. The PL-ACO has a lower cost value than the traditional ACO and ACO-GA method. The PL-ACO has the advantage of replacing low-performance ant with Prime learning ant, which improves the learning rate of the method. The traditional ACO method has 60 iterations for 0.4 cost, and the proposed PL-ACO method has 4 iterations for 0.4 cost value. The learning rate improvement increases the performance of the PL-ACO method more than the traditional method.

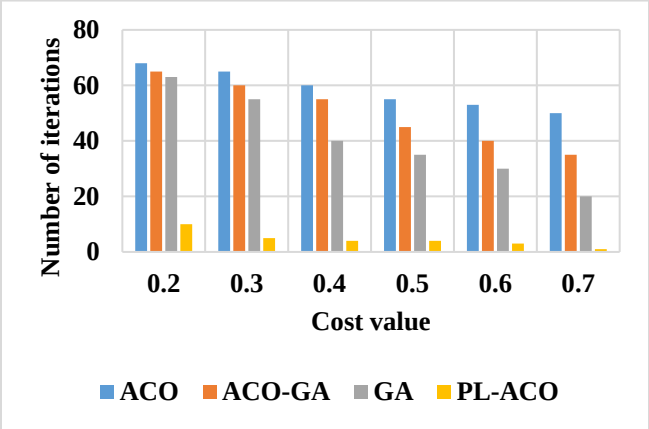

Fig. 2 Number of iterations for various cost levels

Table 3. Plans with a cost of 0.2 required a number of iterations for each method

No. of Plans	ACO	ACO-GA	TLBO	PL-ACO
5	77	75	70	6
10	82	78	73	9
20	93	84	78	11
30	103	94	83	12
40	118	105	88	15
50	126	115	92	15

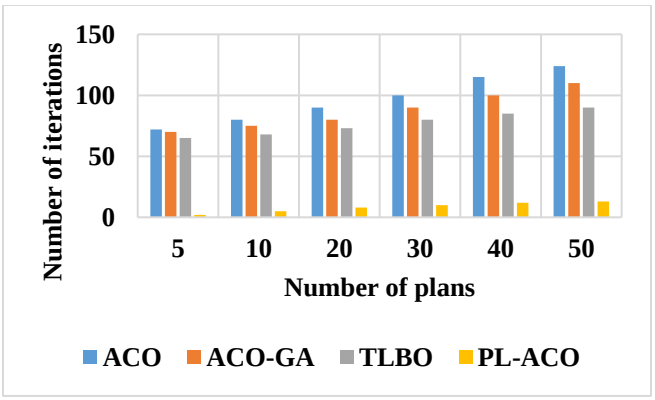

Fig. 3 Number of iterations for various plans with a cost of 0.6

Table 4. Plans with a cost of 0.6 required a number of iterations

No. of Plans	ACO	ACO-GA	TLBO	PL-ACO
5	74	72	68	4
10	83	79	70	9
20	94	85	75	10
30	105	92	84	14
40	120	104	90	16
50	128	114	93	16

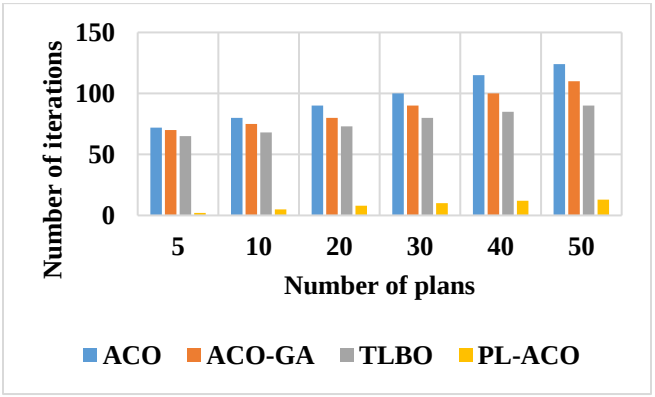

Fig. 4 Plans with a cost of 0.6 required number of iterations

Table 5. Plans with a cost of 0.6 required a number of iterations for each method

No. of Plans	ACO	ACO-GA	GA	PL-ACO
5	75	75	67	7
10	83	80	70	10
20	95	83	78	10
30	103	92	82	12
40	117	105	89	15
50	128	112	95	16

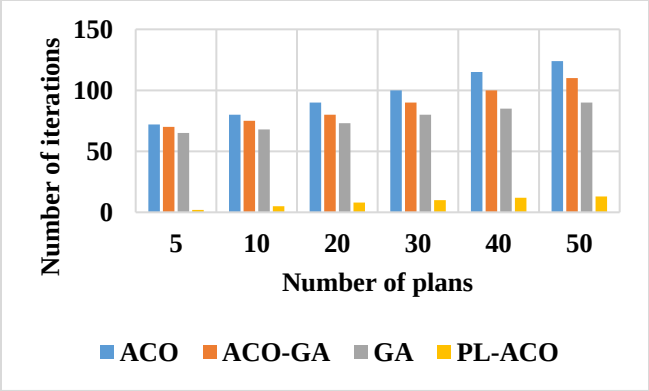

Fig. 5 Number of iterations for various plans with the cost of 0.6

The number of iterations for various plans with a cost of 0.6 for proposed and existing methods are shown in Table 3 and Fig. 3. The number of iterations is less for the proposed method for the various number of plans than existing methods. This is because the proposed method replaces the best learning ants with the best. Easily trapped into local optima, poor convergence affects existing methods' performance.

The number of iterations for various proposed and existing methods with a cost of 0.6 is shown in Table 4 and Fig. 4. The proposed method has fewer iterations than existing methods due to improvement in the learning process. The existing methods have the limitation of local optima and poor convergences that affects model performance.

The number of iterations of the proposed method and existing method for various plans with the cost of 0.6 is given in Fig. 5 and Table 5. The PL-ACO has less number of iterations than existing methods due to its advantage of replacing the worst learner with the best learner. It helps the model to increase convergence rate and escape from local optima of the query optimization.

Table 6. Query cost of PL-ACO method for various iterations

No. of Entity	ACO [16]				QIACO [16]				PL-ACO			
	No. of iterations	Worst Cost	Average Cost	Best Cost	No. of iterations	Worst Cost	Average Cost	Best Cost	No. of iterations	Worst Cost	Average Cost	Best Cost
5	100	0.01341	0.01304	0.01239	100	0.01138	0.01138	0.01138	100	0.00953	0.00908	0.0885
10	100	0.88999	0.352626	0.10301	100	0.23963	0.11929	0.06968	100	0.23121	0.11541	0.05822
15	300	6538.6434	2367.6559	158.8261	100	58.2545	18.6617	2.3471	100	48.4357	16.54782	2.0148
20	300	51574.727	20818.5094	1506.295	100	9689.5597	4627.8784	890.405	100	9420.2549	4421.58963	784.5621

Table 7. Query cost of PL-ACO method for various number of tables

Number of tables	4	6	8	10	12
GA [17]	26.5	73	83.53	107.71	96.64
EGA [17]	26.5	72.55	78.25	96.98	92.84
ADEGA [17]	26.5	70.95	76.31	85.96	81.66
ACO	24.7	68.54	75.82	83.65	78.22
PL-ACO	24.7	67.93	72.34	82.14	77.51

Table 8. Iterations of the PL-ACO method for various number of tables

Number of tables	4	6	8	10	12
GA [17]	105	117	167	178	216
EGA [17]	104	110	139	155	186
ADEGA [17]	102	108	122	145	178
ACO	100	106	118	136	172
PL-ACO	100	102	110	128	164

Table 9. Search time of PL-ACO method for various number of tables

Number of tables	4	6	8	10	12
GA [17]	5.62	11.44	19.61	30.59	42.37
EGA [17]	5.55	9.61	15.32	24.27	28.5
ADEGA [17]	5.38	7.89	12.9	18.67	20.52
ACO	5.24	7.54	10.82	12.56	18.47
PL-ACO	5.10	6.93	9.41	8.29	15.13

The proposed method is evaluated for TPC-H benchmark queries per the criteria of [16] and [17]. The query processing cost of PL-ACO and existing methods are given in Table 6 for the various numbers of entities and iterations. The PL-ACO method has a lower query processing cost due to its ability to create the fitness function for each possible query join helps to find the optimal solutions. The existing method increases the search space to find the solution that increases the query cost of the model. The PL-ACO method maintains the search space based on the join query search in the model.

The PL-ACO method query cost is compared with existing methods such as GA, EGA, and ADEGA, as shown in Table 7. The PL-ACO method has lower query cost due to its measured fitness value for every query joint in the system and limits the search space based on available query joints. The existing method ADEGA [17] measure double entropy for query joints in the system, which tends to increase the query cost of the model.

The PL-ACO method is compared with existing methods for the required number of iterations to find the optimal solutions, as given in Table 8. The PL-ACO method finds the optimal solutions for fewer iterations than existing methods. The PL-ACO method replaces weaker ants with the best fitness ant to increase the learning rate that helps find optimal value with fewer iterations.

The PL-ACO method is compared with existing methods for search time for query optimization, as given in Table 9. The PL-ACO method has less search time than existing methods for query optimization. The PL-ACO method limits the search space based on available query joints in the system, which helps reduce the method's search time. The existing ADEGA [17] method increases the search space and measures two entropy that increases the method's search time.

6. Conclusion

Query optimization helps to improve database efficiency and query cost reduction. The existing methods have applied ACO based query optimization method to reduce the query cost. The existing ACO-based methods have limitations of local optima and poor convergence in the optimization. This research applied the PL-ACO method to improve the performance of query optimization. The system model and the dataset are used to test the PL-ACO method efficiency in query optimization. The PL-ACO method has higher efficiency in query optimization than the existing method. The proposed method provides a lower cost of a query with fewer iterations in query optimization. The PL-ACO method has the advantage of creating fitness value for available query joints that help to find the optimal solution for the query. The PL-ACO method replaces the worst learning ant with the best learning ant to improve the learning rate. The PL-ACO method limits the search space based on available query joins, which helps reduce the system's query cost. The PL-ACO method has an average cost of 16.54 for 100 iterations, and the existing QIACO method has an average cost of 18.66 for 100. The PL-ACO method has 82.14 query costs for 10 tables, and ADEGA has 85.96 query costs for 10 tables. A hybrid optimization method is applied for query optimization to improve query optimisation performance.

References

[1] R. Sahal, M.H. Khafagy, F.A. Omara, "Exploiting Coarse-Grained Reused-Based Opportunities in Big Data Multi-Query Optimization," *Journal of Computational Science*, vol. 26, pp. 432-452, 2018.

[2] M. Renukadevi, E.M. Anita, D. Mohana Geetha, "An Efficient Privacy-Preserving Model Based on OMFTSA for Query Optimization in Crowdsourcing," *Concurrency and Computation: Practice and Experience*, vol. 33, pp. e6447, 2021.

[3] Y. Jiang, K. Zhang, Y. Qian, L. Zhou, "Reinforcement Learning based Query Optimization in Differentially Private IoT Data Publishing," *IEEE Internet of Things Journal*, 2021.

[4] P. Michiardi, D. Carra, S. Migliorini, "Cache-based Multi-Query Optimization for Data-Intensive Scalable Computing Frameworks," *Information Systems Frontiers*, vol. 23, pp. 35-51, 2021.

[5] M. Sharma, G. Singh, R. Singh, "A Review of Different Cost-Based Distributed Query Optimizers," *Progress in Artificial Intelligence*, vol. 8, pp. 45-62, 2019.

[6] L. Chen, Y. Lin, J. Wang, H. Huang, D. Chen, Y. Wu, "Query Grouping–Based Multi-Query Optimization Framework for Interactive SQL Query Engines on Hadoop," *Concurrency and Computation: Practice and Experience*, vol. 30, pp. e4676, 2018.

[7] A. Sebaa, A. Tari, "Query Optimization in Cloud Environments: Challenges, Taxonomy, and Techniques," *The Journal of Supercomputing*, vol. 75, pp. 5420-5450, 2019.

[8] Z.B. Ozger, N.Y. Uslu, "An Effective Discrete Artificial Bee Colony Based SPARQL Query Path Optimization by Reordering Triples," *Journal of Computer Science and Technology*, vol. 36, pp. 445-462, 2021.

[9] E. Mella, M.A. Rodríguez, L. Bravo, D. Gatica, "Query Rewriting for Semantic Query Optimization in Spatial Databases," *Geoinformatica*, vol. 23, pp. 79-104, 2019.

[10] J. Kossmann, T. Papenbrock, F. Naumann, "Data Dependencies for Query Optimization: A Survey," *The VLDB Journal*, pp. 1-22, 2021.

[11] V. Panahi, N.J. Navimipour, "Join Query Optimization in the Distributed Database System using an Artificial Bee Colony Algorithm and Genetic Operators," *Concurrency and Computation: Practice and Experience*, vol. 31, pp. e5218, 2019.

[12] M. Jafarinejad, and M. Amini, "Multi-Join Query Optimization in Bucket-Based Encrypted Databases using an Enhanced Ant Colony Optimization Algorithm," *Distributed and Parallel Databases*, vol. 36, pp. 399-441, 2018.

[13] S.A. Mohsin, S.M. Darwish, A. Younes, "QIACO: A Quantum Dynamic Cost Ant System for Query Optimization in Distributed Database," *IEEE Access*, vol. 9, pp. 15833-15846, 2021.

[14] L. G. Igakeh, V. I. E. Anireh, D. Matthias, "Implementation of Ant Colony Optimization For Call Drop In Gsm Network," *SSRG International Journal of Computer Science and Engineering*, vol. 7, no. 2, pp. 57-67, 2020. *Crossref*, https://doi.org/10.14445/23488387/IJCSE-V7I2P107

[15] D. Kumar, and V.K. Jha, "An Improved Query Optimization Process in Big Data Using ACO-GA Algorithm and HDFS Map Reduce Technique," *Distributed and Parallel Databases*, vol. 39, pp. 79-96, 2021.

[16] S.A. Mohsin, A. Younes, S.M. Darwish, "Dynamic Cost Ant Colony Algorithm to Optimize Query for Distributed Database Based on Quantum-Inspired Approach," *Symmetry*, vol. 13, no. 1, pp. 70, 2021.

[17] B. Zheng, X. Li, Z. Tian, L. Meng, "Optimization Method for Distributed Database Query Based on An Adaptive Double Entropy Genetic Algorithm," *IEEE Access*, vol. 10, pp. 4640–4648, 2022.

[18] W. Deng, J. Xu, and H. Zhao, "An Improved Ant Colony Optimization Algorithm Based on Hybrid Strategies for Scheduling Problem," *IEEE Access, vol.* 7, pp. 20281-20292, 2019.

[19] J. Uthayakumar, N. Metawa, K. Shankar, S.K. Lakshmanaprabu, "Financial Crisis Prediction Model Using Ant Colony Optimization," *International Journal of Information Management*, vol. 50, pp. 538-556, 2020.

[20] M. Paniri, M.B. Dowlatshahi, H. Nezamabadi-pour, "MLACO: A Multi-Label Feature Selection Algorithm Based on Ant Colony Optimization," *Knowledge-Based Systems*, vol. 192, pp. 105285, 2020.

[21] Y. Li, H. Soleimani, M. Zohal, "An Improved Ant Colony Optimization Algorithm for the Multi-Depot Green Vehicle Routing Problem with Multiple Objectives," *Journal of Cleaner Production*, vol. 227, pp. 1161-1172, 2019.

[22] C. Xu, B. Gordan, M. Koopialipoor, D.J. Armaghani, M.M. Tahir, X. Zhang, "Improving Performance of Retaining Walls Under Dynamic Conditions Developing an Optimized ANN Based on Ant Colony Optimization Technique," *IEEE Access*, vol. 7, pp. 94692-94700, 2019.

[23] M.A. Ariffin, R. Ibrahim, I.S. Ibrahim, J.A. Wahab, "Test Cases Prioritization Using Ant Colony Optimization and Firefly Algorithm," *International Journal of Engineering Trends and Technology*, vol. 70, no. 3, pp. 22-28, 2022.

[24] N Bhushana Babu D, E V Krishna Rao, K.S.N. Murthy, "Inter-Gateway Handoff Management Using Ant Colony Optimization (ACO) for Wireless Mesh Networks," *International Journal of Engineering Trends and Technology*, vol. 68, no. 11, pp. 63-71, 2020.

[25] X.Y. Gao, R. Sahal, G.X. Chen, M.H. Khafagy, F.A. Omara, "Exploiting Sharing Join Opportunities in Big Data Multiquery Optimization with Flink," *Complexity*, 2020.