Journal of Theoretical and Applied Information Technology

30th November 2023. Vol.101. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ABC ALGORITHM AS AN ENHANCEMENT FOR MQO PROCESS IN BIG DATA

MANAL A. ABDEL-FATTAH [1], SAYED ABDELGABER [2], S. A. NASR [3], WAEL MOHAMED [4]

[1,2,3,4] Faculty of Computers and Artificial Intelligence, Information system department,

Helwan University, Cairo, Egypt

E-mail: [1] Manal_8@hotmail.com, [2] sgaber14@gmail.com ,
[3] safaa.anwer.nasr@gmail.com, [2] waelmohamed@fci.helwan.edu.eg

ABSTRACT

Multi-query optimization is the task of generating an execution plan for a collection of multiple queries. In recent years, big data querying has become an important field because it provides better data understanding and valuable insight. This paper studies the ability to enhance the process of multi-query using one of the swarm algorithms. Join operation is the most time-consuming operation, the study focuses on join operation and illustrates the effect of the join execution order on the time. Many techniques can be used to decide the optimal order to execute a set of join operations, swarm algorithms are proposed in this research to scan all possible solutions and choose the optimal one. The paper provides a model for the process, examines it on the big data set, and compares it with previous work. The experiment results that applying artificial bee colony algorithm on multi-query optimization enhances the time of execution.
Keywords: *Multi-Query Optimization (MQO), Query Execution Plan (QEP), Artificial Bee Colony (ABC)*

1. INTRODUCTION

Many organizations own a large amount of data, and they tend to analyze it to extract data, gain business benefits, and collect information about customers and their habits.

Big data querying allows businesses to quickly and easily search their data. There are huge chunks of data that aren't relevant to the business, and a lot of that data is useful. It is a hard process even for experts to decide what is valuable for the business and what isn't from this sheer amount of information available. Querying data can separate irrelevant data and let experts focus on useful data. Quiring huge amounts of data is not a normal quiring process, the first challenge of big data querying is that big data is very big, with millions of records about customers' interests, website visitors, conversion rates, churn rates, financial data, and so much more. This multi-query can be duplicated, and conflicted, that cause overhead from large files resulting from every single query.

Performing multiple queries on big data is necessary to extract valuable insights and information from the data. With big data, there is often too much information to analyze in one query, and breaking it down into smaller queries can help to identify patterns, trends, and correlations that may have been missed in a single query. Additionally, multiple queries can help to refine the analysis and ensure the accuracy of the results. By performing multiple queries, organizations can gain a more comprehensive understanding of their data and make informed decisions based on the insights gleaned from the analysis.

Traditional data analysis methods have certain limitations. For example, they may not be able to handle large volumes of data, they may not be able to process data in real-time, and they may not be able to handle unstructured data. Additionally, traditional methods may not be able to identify complex relationships and patterns in the data.

Many researchers [1] [2] [3] [4] [5] handle this problem on a relational database by so many techniques, such as the grouping technique in [6], the Shared Predicate-based technique [7], the greedy algorithm [8], and different swarm algorithms particle swarm [9], such as bee algorithm [10] [11] [12], ant colony [13] [14] and many other techniques than optimize multi-query.

In our research [15], the comparison between different techniques concludes that the artificial bee colony (ABC) algorithm is the most efficient swarm algorithm for dealing with MQO.

In this research, MQO was applied to a big dataset and solved using the ABC algorithm. The research mainly focuses on certain feature of big data which is volume. The research illustrates the ability of enhancing the process of querying bigdata. The research also focuses on using one of swarm

algorithms to perform this enhancement. The enhancement intended in this research is the increasing of the time needed to execute the multi-query. The swarm algorithm used in this research determined based on a comparison conducted between different swarm algorithms [15] to find the most appropriate one for our case. The experiment performed on big data set to evaluate algorithm and compare it with previous works. Large scale data is our scope, other big data features are out of the scope of this research.

The rest of this paper is organized as follows. We start in section 2 with formal definitions of multi-join optimization, why we need to optimize multi-query, and why we focus on join operation. We discuss ABC Algorithm as a solution for MQO on big data in Section 3, the technique has been discussed, and pseudocode has been illustrated. In Section 4, the experiment has been conducted, discussed, and the results have been summarized. Also, a comparison has been conducted between multi-query ABC technique and technique used in [16]. Finally, Section 5 concludes the paper

2. MULTI-JOIN OPTIMIZATION

The optimization of the query response strategy is an area where improvements can be made. This is thus a major step in the processing of queries. Processing a query involves a variety of activities that are required to retrieve information from the database. These activities contain the queries translation into suitable expressions that can be implemented at the file system's level since these queries are delivered to the DBMS in a high-level language, query optimization steps, transformations, and query evaluation. Multi-join Query optimization is a complex problem, not only in the SQL server but in any other relational database system.

In case of a single query at a time, the executor generates a temporary file to hold the result of the query. If we do the same on the multi-query, we will create excessive overheard, time-consuming, and unnecessary files in many cases because these files will be used as input to the next operation. This is where the importance of generating the query execution plan that corresponds to algorithms for a combination of operations in a query. For example, we have 4 relations with 3 join operations, as shown in Figure 2. Instead of creating three temporary files, we apply the plan directly and get just one result file. The size of the file that holds the result increases as the data volume is increased. So, dealing with huge files in big data forces us to use a technique to reduce duplication or any overhead that becomes extremely big.

If the user inputs a query, the query is first analyzed by the parser for syntax errors, and if there is no error, the query is converted to a standard format, query graph [9]. Next, the query optimizer is taking this query graph as an input and setting up different query execution planes for that query according to which it will be best performed by selecting optimal query execution plans among all these. This optimal query plan shall then be transferred to a query execution engine in order to determine its performance and return the query result

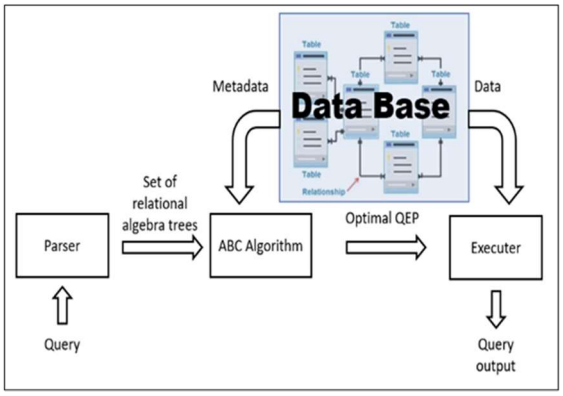


Figure 1: Query Evaluation And Answering Process And The Role Of ABC Algorithm In This Process

Individual queries are transformed into relation algebra expressions (algebra trees) and are represented as query graphs. After that, a query optimizer has selected an appropriate Physical Method for implementing each relational Algebra operation and thus produced the query execution plan. For all equivalent QEPs, the optimizer selects a query execution engine with the lowest cost output to perform an operation on that plane and returns answers to the user. The process showed in Figure 1

2.1 The order of executing join operation

Many factors affect the optimization process depending on the procedural behavior of executing the query such as the shape of the join query and the order of the join operation. This effect is increased when applying the join process to big data. Also, if data is distributed over multiple sites, the allocation of data and the arrangement of queries among these sites affect the optimization process. The most influential factor is the order of join operation of the involved relations in the query. The join operation is one of the most time-consuming operations in query processing.

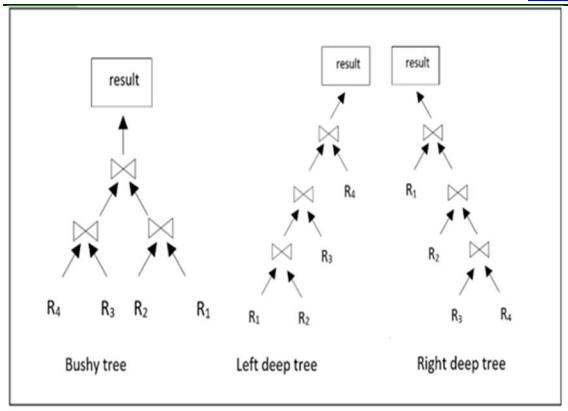


Figure 2: Different Shapes Of Relational Algebra Trees

In relational database systems, each query execution plan can be represented by a processing tree where the leaf nodes are the base relations, and the internal nodes represent operations. Different tree shapes have been Considered: left-deep tree, right-deep tree, and bushy tree. Figur.2 explains tree structures of relational operators associated with the multi-join query R1∞R2 ∞R3∞R4.

Search space can be restricted according to the nature of the execution plans and the applied search strategy. The nature of execution plans is determined according to two criteria: the shape of the tree structures (i.e. left-deep tree, right-deep tree, and bushy tree) and the consideration of plans with Cartesian products.

For every relational algebra tree shape that contains n of relations involved in the join operation, there is n! of different possible arranges. In our example that contains only 4 relations we have 24 arranges for every algebra tree shape, the optimizer has to handle 72 different arranges and find the most optimal one.

2.2 Evaluating QEP

The solution space of the MJQO problem is the set of all possible join processing trees for a query graph (Query Execution Plans). The goal is to find out the join ordering tree that has a minimal cost in the mentioned solution space at the best time.

Each relation in the query graph required parameters which are: n(r): number of tuples in relation r; v (A, r): number of distinct attributes as in relation r. The formula to calculate the cost of a join tree execution is [9].

$$cost = \sum_{i=1}^{n-1} n(t_i) \quad (1)$$

$$n(t) = \frac{n(r) + n(s)}{\max\big(v(c,r), v(c,s)\big)} \quad (2)$$

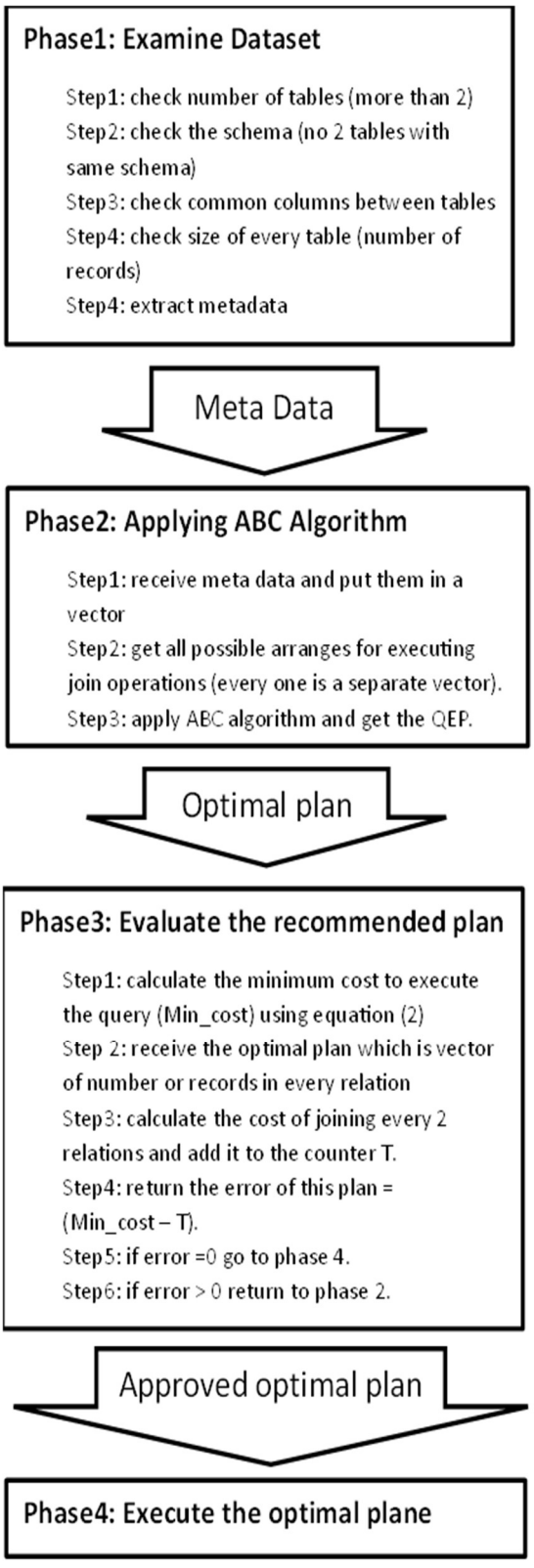


Figure 3: Model For Multi-Query Enhancement Process

For the inner node, if r and s are relations represented by respect timely by the left child and the right child of t, and C is a common attribute group in relation, then n (t) is the size result relation of the join operation of tow relation r and s; which is equal to the number of rows having similar values of attribute common in both relation, r and s.

2.1 Nature of data

Our scope in this research will be large-scale structured data. The nature of data is an important factor, that makes it possible to conduct the join operation on it. Existing of common columns between tables is a main restriction.

Figure 3 shows the phases of the technique to apply ABC algorithm on large-scale dataset. Examining data is a pre-step to find a common column between tables and collect metadata that interred to the ABC algorithm as an input. ABC algorithm creates the solution area and suggests an optimal plan, then the error function tests this plan and retrieves approval plan to be executed.

3. ABC ALGORITHM AS A SOLUTION

As discussed, our goal is to find the most optimal QEP that answers the query with a minimum time of execution. The proposed approach to handle this problem is to apply a swarm algorithm on a solution set to find the optimal solution. According to our research [11], the most suitable swarm algorithm is Artificial Bee Colony (ABC).

The Artificial Bee Colony (ABC) is one of the swarm algorithms, which is proposed to be applied on MQO. The beauty of ABC lies in the selection and neighbor production mechanism. This becomes the prime reason behind the efficient working of ABC. It works well for both local and global optimization.

Any bee that is currently exploiting a food source shall be known to have been employed. The worker bees manipulate the food source, keeping the information about the food source in the hive and sharing it with the observer bees. The observer bees will wait for information to be provided by employed bees on their discovery of food sources, while scout bees would seek out new feeding areas close to the hive.

3.1 Mapping the algorithm

The algorithm started by randomly generating a solution set which is represented as a set of vectors in our case instead of nodes in the original algorithm. Every employed bee is associated with a single solution and evaluates its quality by calculating the fitness function (3), then sharing this information with onlookers. Onlookers in turn compare the solutions they get information about by using the probability function.

$$p_i = \frac{f(x_i)}{\sum_{k=1}^{s} f(x_k)} \qquad (3)$$

Algorithm1: Modified Artificial Bee Colony	
1	Initialization
2	The population: a set of vectors containing the number of records in every relation
3	Set NumIter: number of iterations
4	Iteration ← 0
5	Improvement
6	do while (iteration < NumIter)
7	for i=0 #Employee
8	NewQEP ← select random QEP
9	NewError ← NewQEP error
10	if NewError < BestError
11	BestError ← NewError
12	BestQEP ← NewQEP
13	else
14	Find random QEP
15	Determine probability for NewQEP $p_i = \frac{fitness_i}{\sum fitness_i}$
16	end for
17	for i=1 # onlookers
18	bestSol ← solution with a higher probability
19	RandomSol ← apply a random solution
20	if (RandomSol < BestSol)
21	BestSol ← RandomSol
22	end if
23	end for
24	Scout bee checking for exploited solutions and replace them with new solutions
25	iteration++
26	end do

Where k is a random number defined as k∈(1, 2, …, s), k≠0, s is the number of solutions in the solution area which is equal to the factorial of the number of relations involved in the query,

Model for multi-query enhancement process and f(xi) is the fitness function calculated for every solution (equation 1), and it measures the quality of this solution located (xi) by calculating the time of execution and the error of this solution.

Definition-- the ith solution represented as xi= (xi1, xi2, …, xid), where d is the number of relations involved in the query.

Onlooker bees will be able to determine from the scans which of them comes up with the best solution based on the earlier equation. With every comparison the onlooker performed, they also calculate the error and compare it with the best solution's error, the error of now solution = (fitness of my solution – the best fitness cached). This way the agents can recognize if they are far or near to the optimal solution. ABC algorithm pseudocode is in Algorithm1.

4. EXPERIMENTAL RESULTS AND DISCUSSION

In This section we will describe the nature of the dataset used for testing and evaluating our algorithm, the computational resources used to implement and apply the technique, and discuss the results we get.

The proposed technique has been evaluated by applying it to a big dataset, with about 9 million total number of records, 12GB total size, and 14 tables. The data is about fitness and daily activity hours and their effect on heart rate, calorie loss, nature of sleep, and so on.

4.1 Experimental Setup

The experiment in this research was conducted on a computer with processor coreI7-4810 MQ 2.8 GHz and 16 GB RAM. ABC algorithm has been implemented using Python programming language (pyspark library) on Anaconda Navigator (Spyder) software.

The algorithm is designed to optimize a large number of QEPs answers multi-query over a big dataset with a minimum time of execution. The agents in this algorithm search, construct, and evaluate the different join orders and retrieve the order that has a minimum response time. The algorithm receives queries and metadata about tables involved in these queries as input and then retrieves the optimal answer to be executed.

4.2 Discussion

Table1. and Figure4. Summarize the results of performing different numbers of multi-query processes from 4 to 10 joins. Applying the ABC algorithm enhances the stability of execution time, the huge difference in the size of data doesn't make a huge swing in the time of execution. On the contrary, when conducting ordinary join, we observed that the size of data makes a significant effect on execution time. Figure3. Shows the time of executions in milliseconds in both cases. We can see that applying ABC gets close results even with the increasing number of joins, that accordingly, increased size of data resulted.

TABLE1: Results of applying ABC join and performing ordinary join on large-scale dataset

No. of Query	No, of Iterations	Ordinary Join		ABC Join	
		Best	Average	Best	Average
4	6	69304	97970	7503	7756
6	10	87057	160400	7780	7866
10	10	202777	315146	8329	8778

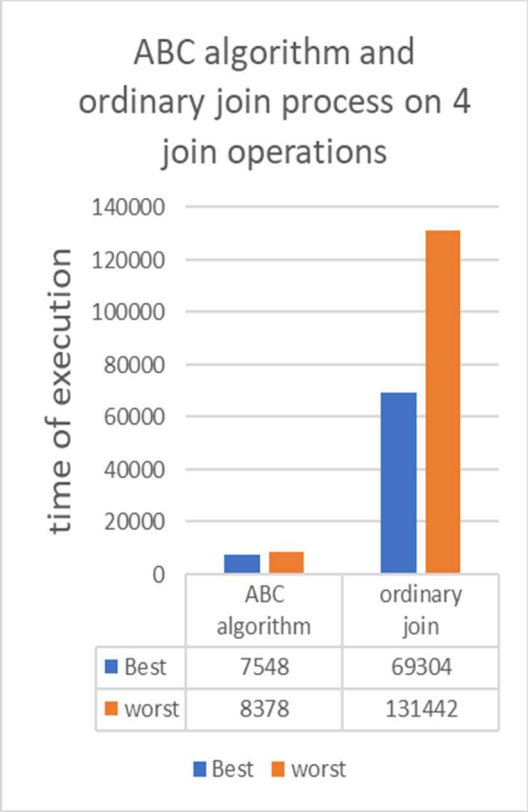


Figure 4: The best and the worst time of execution in case of applying 4 join operations

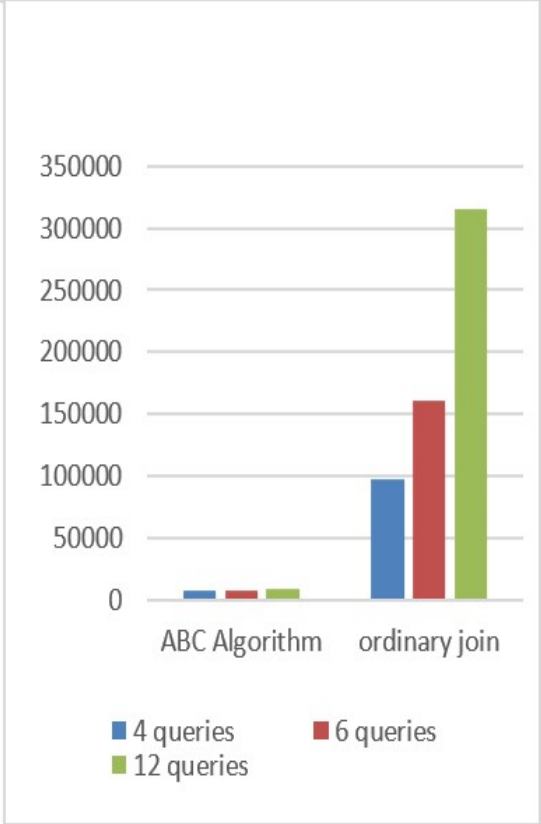


Figure 5: The effect of data volume on join operation execution efficiency

In this experiment, every multi-query has been conducted many iterations. In figure 4 for example we get the best and the worst time of execution, and then in figure 5 results of all iterations have been summarized in best and average time of execution.

The first multi-query examined was contains 4 queries, and it has been conducted six times. The ordinary join execution costs in average 97 seconds, and the best time recorded was 69 seconds. The 6 queries in best cases were executed in 87 seconds, and the 10 queries execution arrived to more than 300 seconds to get the results.

On other hand, when using ABC algorithm on the same data set and the same set of queries, we get results between 7 and 8.7 seconds only. Also, the stability of execution time is an observed point when applying ABC algorithm. Because of the technique used on ABC algorithm that based on finding the shortest way to execute the given query with no need to save a huge temporary file that slow the process of executing query.

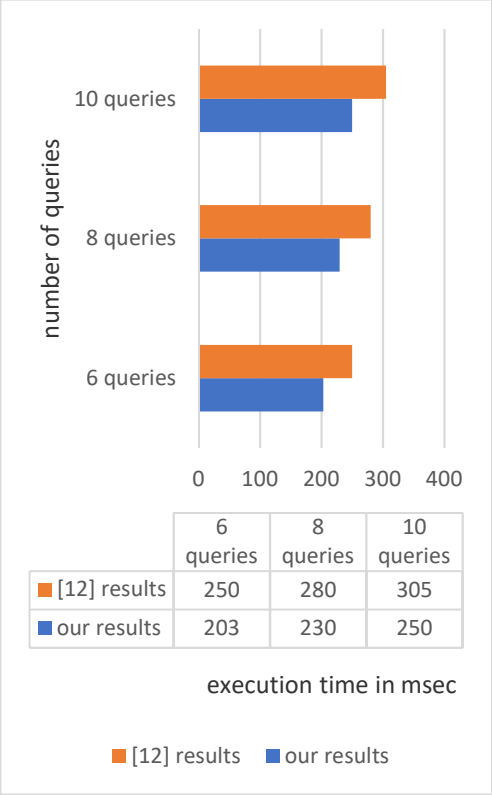
4.3 Comparison with previous work

In [12], the authors propose to improve multi-join query process by using parallel computing in two phase optimization strategy. The paper discusses the optimization of multi-join queries, which are important operations in data management and integration systems. Multi-join queries aggregate data from multiple tables or data sources to provide material for applications such as data integration, data sharing, and decision support. The efficiency of multi-join queries is crucial for real-time and effective use of data, and is affected by various factors such as the number of joins, the amount of data, the join selectivity, the execution order of joins, the storage location of data, and the resources, strategies, and methods used for parallel optimization. The paper focuses on using GPU technology to optimize multi-join queries and proposes a multi-phase optimization strategy and optimization methods for each stage. The experimental results show that the proposed optimization algorithm improves the efficiency of

multi-join queries, especially in the case of high load and complex join queries, achieving higher throughput than previous optimization algorithms.

The authors apply their method to the TPC-DS benchmark. TPC-DS data set consists of 24 tables with more than 19 million records. To evaluate the ABC algorithm, we apply it to the same data set to compare the time of executing under a fixed factor (queries and dataset). The experiment selects queries with more join operations (than other operations). From results summarized in figure 6, ABC algorithm provide an enhancement on execution time.

The comparison performed in three sizes of multi-query, which are 6 queries, 8 queries, and 10 queries.



	6 queries	8 queries	10 queries
[12] results	250	280	305
our results	203	230	250

Figure 6: The execution time of join operation on the same data under ABC algorithm and [12] technique

Analyzing figure 6 results that in case of 6 queries the enhancement was 18.8%, in case of 8 queries the enhancement was 17.9%, and in case of 10 queries the enhancement was 18% better than paper [12].

5. CONCLUSION

The paper proposed (Artificial Bee Colony) ABC algorithm as an enhancement for (Multi-Join Query Optimization) MJQO process applied on big data. The paper discussed the ABC algorithm and modeled it on the MJQO process.

The importance of ordering in executing the join operation has been discussed, and the equations used to evaluate QEPs have been previewed. Also, the paper proposed a model for the process of applying the ABC Algorithm to enhance the time of executing multi-query, and it introduced the algorithm that described the process.

The multi-query ABC algorithm has been applied to a big dataset and compared its performance with conducting the same queries on ordinary join. The results showed a large difference in the performance of the two techniques in favor of the proposed technique. The main improvement is the enhancement of quiring process on bigdata set which was usually performed on relational database only.

The paper also compared the proposed technique with previous work in the same area. The comparison results that ABC accomplished good enhancement in the time of execution.

This research doesn't debate other bigdata features, so authors recommend for future work applying the same technique to other features of big data doesn't tackle in this research. They recommend for example, enhancing distributed data querying in case of applying multi-query on distributed datasets.

References

[1] A. Jonathan, A. Chandra, and J. Weissman, “Multi-query optimization in wide-area streaming analytics,” *SoCC 2018 - Proc. 2018 ACM Symp. Cloud Comput.*, pp. 412–425, 2018, doi: 10.1145/3267809.3267842.

[2] A. K. Z. Al Saedi, R. B. Ghazali, and M. B. M. Deris, “An efficient multi join query optimization for DBMS using swarm intelligent approach,” *2014 4th World Congr. Inf. Commun. Technol. WICT 2014*, pp. 113–117, 2014, doi: 10.1109/WICT.2014.7077312.

[3] H. A.Hanafy and A. M. Gadallah, “Ant Colony-Based Approach for Query Optimization,” *Data Min. Big Data*, pp. 425–433, 2015, doi: 10.1007/978-3-319-40973-3.

[4] A. K. Z. Alsaedi and M. M. D. Rozaida Ghazali, “An Efficient Multi Join Query Optimization for Relational Database

Management System Using Two Phase Artificial Bess Colony Algorithm," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9429, pp. 443–453, 2015, doi: 10.1007/978-3-319-25939-0.

[5] A. K. Z. Alsaedi, R. Ghazali, and M. M. Deris, "An efficient multi join query optimization for relational database management system using two phase artificial bess colony algorithm," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9429, pp. 213–226, 2015, doi: 10.1007/978-3-319-25939-0_19.

[6] L. Chen, Y. Lin, J. Wang, H. Huang, D. Chen, and Y. Wu, "Query grouping–based multi-query optimization framework for interactive SQL query engines on Hadoop," *Concurr. Comput.* , vol. 30, no. 19, pp. 1–16, 2018, doi: 10.1002/cpe.4676.

[7] R. Sahal, M. H. Khafagy, and F. A. Omara, "Comparative study of multi-query optimization techniques using shared predicate-based for big data," *Int. J. Grid Distrib. Comput.*, vol. 9, no. 5, pp. 229–240, 2016, doi: 10.14257/ijgdc.2016.9.5.20.

[8] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe, "Efficient and extensible algorithms for multi query optimization," *SIGMOD Rec. (ACM Spec. Interes. Gr. Manag. Data)*, vol. 29, no. 2, pp. 249–260, 2000, doi: 10.1145/335191.335419.

[9] S. Lalwani and H. Sharma, "Multi-objective three level parallel PSO algorithm for structural alignment of complex RNA sequences," *Evol. Intell.*, vol. 0, no. 0, p. 0, 2019, doi: 10.1007/s12065-018-00198-y.

[10] M. Alamery, A. Faraahi, H. H. S. Javadi, S. Nourossana, and H. Erfani, "Multi-join query optimization using the bees algorithm," *Adv. Intell. Soft Comput.*, vol. 79, pp. 449–457, 2010, doi: 10.1007/978-3-642-14883-5_58.

[11] M. Alamery, A. Faraahi, H. H. S. Javadi, S. Nourossana, and E. Hossein, "Application of Bees Algorithm in Multi-Join Query Optimization," *Adv. Comput. Sci. An Int. J.*, vol. 1, no. 1, pp. 5–9, 2012.

[12] W. Sun, M. Tang, L. Zhang, Z. Huo, and L. Shu, "A survey of using swarm intelligence algorithms in IoT," *Sensors (Switzerland)*, vol. 20, no. 5, 2020, doi: 10.3390/s20051420.

[13] D. Kumar and V. K. Jha, "An improved query optimization process in big data using ACO-GA algorithm and HDFS map reduce technique," *Distrib. Parallel Databases*, vol. 39, no. 1, pp. 79–96, 2021, doi: 10.1007/s10619-020-07285-z.

[14] A. Nayyar, "Ant Colony Optimization-Computational Swarm Intelligence Technique," pp. 1493–1499, 2016.

[15] S. Abdelgaber, M. A. Abdel-Fattah, and S. A. Nasr, "Why Bee colony is the most suitable with multi-query optimization ?," pp. 1–6.

[16] X. X. Hu, J. Q. Xi, and D. Y. Tang, "Optimization for Multi-Join Queries on the GPU," *IEEE Access*, vol. 8, pp. 118380–118395, 2020, doi: 10.1109/ACCESS.2020.3002610.