ISSN 2278-3091
Volume 8, No.5, September - October 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse87852019.pdf
https://doi.org/10.30534/ijatcse/2019/87852019

Impact of Varying Parameters in Improved Multi Colony Ant Optimization Algorithm for Join Order Problem in Distributed Databases

Anjali Soni[1], Dr. Swati V. Chande[2]
[1]Research Scholar, Rajasthan Technical University, Kota, Rajasthan, India, anjalisoni81@gmail.com
[2]Professor, Computer Science, International School of Informatics and Management, Jaipur, India, swatichande@rediffmail.com

ABSTRACT

Query Optimization in Distributed Databases is a challenging task due to the rising size of the database. The join operation has a great significance for the execution of queries containing multiple relations. An optimizer may suggest many join order for a single query and selects the best join order that executes the query in minimum time. The current optimizers are working well with a lesser number of relations but as the number of relations in a query increases, execution time also increases. To reduce this execution time, optimizers are required to redesigned or altered. In this paper, an improved existing algorithm is described to deal with the join order problem in Query Optimization and is based on the Multi Colony Ant Optimization algorithm. The performance of the proposed algorithm depends on various parameters such as the number of iterations, number of colonies, and Ant ratio. In this paper, the impact of these parameters on an improved algorithm is also described.

Kye words: Ant Ratio, Colonies, Iterations, Response Time

1. INTRODUCTION

A distributed database collects data from multiple sites to answer any query. Data stored at these sites may be replicated or in fragmented form. To execute a query in a distributed environment, data are accessed from different sites. The Query Processor translates the high-level query into a relational algebra query. There may be many relational algebra plans for a single high-level query. The Response Times for the execution of these plans are very much different. In the distributed environment, the query processing is done on all the relevant sites that include optimization at both the global and local level [1]. Query Optimizer is responsible for generating different execution plans that may have different Response Time and chooses the best one.

A Query Optimization Problem is a difficult problem. It needs a Search Space (possible plans), Fitness Function (Cost Estimations Technique) and an algorithm that finds the best plan from available search space [2]. As the number of relations participating in the query increases, it increases the execution time of a query. The role of the Query Optimizer is to select an effective plan that minimizes the execution time. Therefore, if an effective plan is chosen for the execution of the query, the result of the query will be produced faster. To find out the solutions to the query that have more than one relation, relations must be joined first. To find the best execution plan, an optimizer generates different join orders. The optimizer selects the join order plan that has the minimum Response Time.

In the present scenario, the size of the data dispersed on various nodes in a distributed environment is increasing and the query submitted in Distributed Environment involves a large number of relations. Several Query Optimization algorithms have been proposed and they are giving encouraging solutions [3][4][5][6]. However, due to the increasing number of relations in a join query, the algorithms that have been designed to deal with the problem of query optimization need modifications. Therefore, an optimizer is required to deal with a large number of relations in the join query.

2. REVIEW WORK

From the past many years, different algorithms have been working on Query Optimization Problem. Deterministic strategies work well for the number of joins less than 15 but for a larger number of joins, the time complexity is increased. For a larger number of queries, Randomized Strategies work better, but due to the in-deterministic nature of Randomized Algorithms, the execution time can never be predicted [3]. The problem with Dynamic Programming is high memory consumption for the storage of partial solutions. The features of an Evolutionary Algorithm include parallelism, positive feedback mechanism, negative feedback mechanism [7], and the capability to deal with the query optimization problem. Genetic Algorithm, PSO, ABC, ACO, belongs to the category of Evolutionary Strategies. The problem with GA is that optimal solutions are not guaranteed. The PSO can converge prematurely [4] and ABC may give premature local solution [5].

As discussed above, many algorithms are already working on the Query Optimization problem, but have some limitations. It motivates to improve an existing algorithm that produces a plan to a given query in a minimum Response Time. Ant Colony Optimization Algorithm works well for various Optimization Problems including Query Optimization problem [6]-[10]. In an ant colony, ants search their food and follow a path leaving pheromones. Other ants follow the path led by the ants based on pheromone values deposited by the ants. Ants find an optimal solution from their source to the destination by taking advantage of the knowledge of pheromone trails. [11][12].

ACO helps in finding the solution to the join query, but it suffers from premature convergence [13]. After analyzing the application of ACO for different problems, many variations have been suggested. One of these variations is the method that implements ACO in parallel to better utilize its properties [7][9]. A variation of ACO, where several colonies try to solve the problem simultaneously is known as Multi Colony Ant Optimization (MCAO). MCAO algorithm is an extension of ACO that works with multi ant colony instead of a single ant colony, in parallel. It exchanges the pheromone information between colonies that helps in avoiding the premature convergence problem of ACO. In the past many years, the algorithm has been widely used for many problems such as Unmanned Aerial Vehicle (UAV) path planning [14], dynamic Travelling Salesman Problem [15], Quadratic Assignment Problem [16], Location Routing Problem [17] and has been compared with ACO. It has been analyzed that the performance of MCAO is better than ACO [14]-[16].

The parallel implementation of ACO was recommended in the form of the modified Ant colony [18][19]. In MACO-AVG [20] multi ant colonies were implemented to solve combinatorial optimization problems and it was analyzed that multi colony outperforms single colony. The Parallel ACO [21] was applied to the problem of query optimization and was compared with ACO, Particle Swam Optimization (PSO), and Artificial Bee Colony (ABC). After analyzing the results received from experiments it was observed that Parallel ACO gives better results for join order problem.

Various algorithms are working for the Query Optimization problem but for the increasing number of relations, an optimizer is required that gives an optimal join order to execute the query in minimum time. In this paper, an improved optimization algorithm is proposed to find out the solution to the multi join query in minimum time.

3. PROPOSED ALGORITHM

The objective of creating an optimization algorithm is to generate a plan through available search space that minimizes the fitness function. In the Multi Colony Ant Optimization algorithm, the parameter values are exchanged between multiple colonies of a site. In the proposed algorithm, the pheromone values are exchanged between colonies as well as within sites to improve the quality of solutions produced by different sites. The local pheromone update was applied to avoid the problem of local convergence. The MCAO is improved by applying the global pheromone update method to the MCAO problem i.e. the pheromone values are exchanged between multiple colonies as well as among multiple sites. The global pheromone update shares the pheromone information among multiple sites to improve the output produced by MCAO.

In this research, Improved Multi colony ant optimization is proposed for the solution of the Query Optimization problem in Distributed Databases, and named as MCAA-QODD (Multi Colony Ant Algorithm for Query Optimization in Distributed Databases).

In MCAA-QODD, the Query Generating Site (QGS) is the site where a query is received initially. The query is distributed uniformly to apply join on four sites due to full replication. At each Receiving Site (RS), multiple colonies of ants find the optimal join order and exchange the pheromone values to obtain better solutions. After generating an optimal join order at each RS, local optimal solutions are then sent back to Query Generating Site where again multiple colonies can find the optimal join order. The pheromone values of best colonies that generate the optimal join order are also exchanged between different sites to achieve the global optimal solution.

4. EFFECT OF PARAMETERS ON PROPOSED ALGORITHM

The output of the MCAA-QODD algorithm depends on different parameters such as Information heuristic factor (α), Expectation heuristic factor (β), Pheromone evaporation factor (ρ), Pheromone Strength (Q), Number of Ants, Number of Iterations, Number of Colonies, etc. These parameters have a significant range of values that affect the performance and efficiency of MCAO. The value of Information heuristic factor (α), Expectation heuristic factor (β), Pheromone evaporation factor (ρ), and Pheromone Strength (Q) are adopted by Multi Colony Ant Optimization [15][22]. The parametric setting of the number of ants, iterations, and colonies are determined by analyzing the different values of these parameters on the join order problem.

To determine the value of these parameters, the experiments are conducted in .net framework 4.5 under Visual Studio 2010 Package. C++ language is selected for simulation of the algorithm due to its object-oriented approach. Queries with a different number of relations are then given as input to this algorithm and then the comparison is made with another existing algorithm. The fitness function returns the cardinality of path traversed by ants that perform join operation on all relations on that site also called as cost. For every number of relation, a decision is made based on the value of Response Time and cost. The query is generated in a fully replicated environment.

The MCAA-QODD generates the join order in minimum Response Time with minimum cost. The number of sites assumed is four in Distributed Environment. The optimizer will give the join order of relations from all the four sites simultaneously. For distribution of relations, a query must have minimum nine relations as the distribution of relation is done equally by dividing the number of relations by four if the number of relations are multiple of four. Otherwise, modulus function is applied to number of relations for distribution. If the number of relations is less than or equal to eight then distribution of relations to different sites is not required. For example, if the number of relations in a join query is eight, then the distribution of relations on all four sites will be 2, 2, 2, 2 on each site respectively. There is no need to find out the join order for two relations on each site. Therefore, minimum nine relations are considered for distribution of relations on sites.

In the other case, when number of relations are less than or equal to eight, the join order will be find out only at Query Generating Site (QGS). In this case, two conditions may hold. First, if the number of relations is less than or equal to four, the join order will be find out for four relations in easy manner. Second, for number of relations greater than four in a query, the relations will be considered in a pairs. For example, if the number of relations in a join query is seven (Rel1, Rel2, Rel3, Rel4, Rel5, Rel6, Rel7) then first two relations will form a pair showing the order of join as Rel1 join Rel2. Likewise, Rel3 and Rel4 will form a join order. Moreover, Rel5 and Rel6 is the another pair. At QGS, the join order will be find out for these pairs and the last remaining relation Rel7. For these three pairs and last relation, there will be total 4! possibilities. The optimizer at QGS is responsible for finding the join order with minimum cost. To check the quality of the result, Response Time, as well as cost, is recorded for relations 8, 12, 16……….120.

To implement the MCAA-QODD, impact of various parameters is analyzed. In this paper, total four experiments are shown that contribute for the selection of specific values from the range of varying values of parameters and for the verification of algorithm by comparing it with other existing algorithm. The first experiment focuses on to find out the ratio of ants that gives an optimal output in minimum time. The second experiment finds out the number of colonies in MCAA-QODD that produces the join order in minimal time. The third experiment focuses on to find out the number of iterations that produces an optimal join order in minimum time. The fourth experiment records the Response time for the different number of relations in a query and then compare it with the existing Query Optimization algorithm. The Response Time is recorded by taking into consideration the elapsed time in the selection of join order out of n! available options.

4.1 Experiment 1: Ant Ratio

The first experiment of MCAA-QODD was implemented to find the number of ants required for execution. The experiment is carried out with four ratios of ants in contrast with relations. The experiment was done on the different percentages of ants with the number of relations i.e. 25%, 50%, 75%, and 100%.
Each ratio of ants is executed twenty-five times with the number of joins from 9, 12, 16, …., 120. The Response Time is recorded with each set of ants and relations. Before deciding on the selection of this ant ratio, the cost is also calculated which helps in deciding the ant ratio.

Table 1: Response Time for different Ant Ratios

Ants Percentage/ Relations	100	75	50	25
8	.00848	.00788	.0172	.0148
9	0.04224	0.03172	0.022	0.0232
12	0.05024	0.0318	0.02404	0.0248
16	0.047	0.04452	0.0232	0.0224
20	0.0528	0.0472	0.0232	0.0242
24	0.05788	0.05392	0.0272	0.02772
28	0.0718	0.04944	0.02764	0.03868
32	0.08236	0.07508	0.06432	0.06108
36	0.1032	0.07824	0.0672083	0.06328
40	0.1332	0.10756	0.07892	0.06768
44	0.16892	0.10584	0.1069167	0.0692
48	0.22668	0.14752	0.11448	0.074
52	0.2924	0.14796	0.13976	0.07928
56	0.37916	0.23208	0.19104	0.09128
60	0.48144	0.31712	0.2264	0.1028
64	0.64548	0.40188	0.29964	0.14844
68	0.72812	0.4816	0.35396	0.17496
72	0.88432	0.61248	0.47224	0.2044
76	1.28876	0.78168	0.55408	0.23196
80	1.30204	0.93512	0.72292	0.32976
84	1.6686	1.11412	0.80912	0.38288
88	1.9198	1.35104	1.01912	0.44272
92	2.3846	1.6264	1.14936	0.48532
96	2.92752	1.93216	1.44864	0.66768
100	3.17836	2.24048	1.62712	0.77304
104	4.0672	2.65396	1.9974	0.84388
108	4.6566	3.11916	2.23948	0.93976
112	5.23868	3.66032	2.70552	1.22152
116	6.04592	4.068	2.96496	1.33944
120	6.61092	4.76536	3.59856	1.52564

It can be observed with the above Table 1 that the one-fourth ant ratio gives the optimal join order in minimum time. However, before deciding on the selection of this ant ratio, the cost is also calculated that shows the higher values for the one-fourth ant ratio. Table 2 shows the values received from cost for executing the query for finding out the optimal join order with different ant ratios.

Table 2: Cost for different Ant Ratios

Ant Ratio / Relations	25	50	75	100
4	431	354	435	559
9	460	435	469	451
12	485	430	302	369
16	599	504	575	805
20	953	573	896	1023
24	830	739	697	896
28	1060	837	769	894
32	1225	965	907	963
36	1606	1309	1078	1147
40	1820	1118	1137	1253
44	1907	504	1618	1694
48	2373	1587	1631	1594
52	2314	1927	1729	1802
56	2662	1641	1859	1945
60	2500	1580	1678	1866
64	2550	2096	2247	2315
68	3043	2643	2638	2723
72	3399	2647	2741	2840
76	3571	2905	3200	2922
80	3520	3223	3166	3232
84	4454	3319	3353	3638
88	3792	3360	3382	3715
92	4183	3516	3744	3797
96	4412	3387	3673	3797
100	4747	3667	3724	3908
104	5282	3572	4392	4413
108	5355	3698	4080	4543
112	5344	4659	4668	4882
116	5756	5140	5144	5567
120	6222	4810	5183	5758

A line graph is shown in Figure 1 to show the cost achieved with each ant ratio.

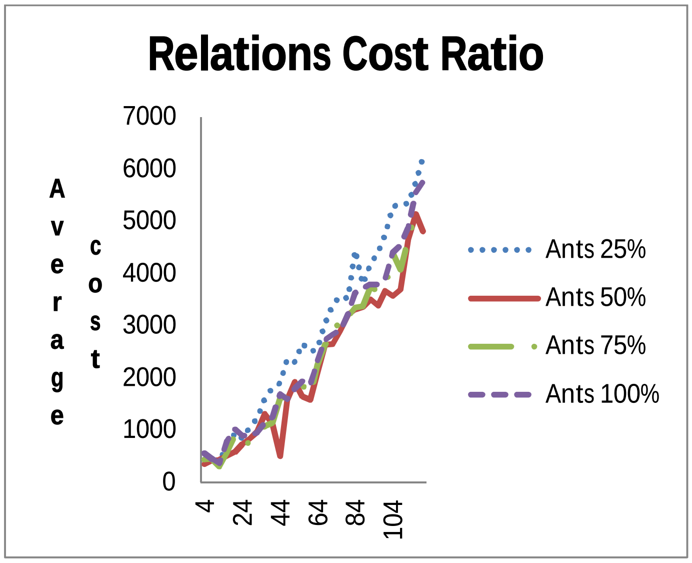

Figure 1: Fitness value (Cost) for Ant Ratios

It can be observed by the values taken from the experiments that the cost decreases by increasing the ant ratio from 25% to 50%. It again increases with an increasing ant ratio from 50% to 75% and again from 75% to 100%. Ant ratio 50% are giving the minimum cost for almost each join query containing relations 4,8,12,……,120 that is better than 25% ant ratio. So in this experiment, the ant ratio has been fixed to 50%.

4.2 Experiment 2: Number of Colonies

After distributing the query on each site by QGS, each site produces the local solution. At each site, the processor has multiple cores [23][24]. The MCAA-QODD is based on multiple colonies so the number of colonies affects the performance of MCAA-QODD. These colonies work on different cores on these processors on each site. So making a decision on number of colonies will affect the number of cores working on that processor. The second experiment of MCAA-QODD was carried out to decide on the number of colonies in MCAA-QODD. Four inputs were taken into consideration i.e. 2 colonies, 4 colonies, 6 colonies, and 8 colonies. After conducting experiments on different sets of relations in a join query, it can be analyzed from Table 3 that four colonies are generating results in minimum time for most of the occurrences in the table. However, the cost is the same for all colonies. Therefore, four colonies were taken into consideration as it produces an optimal join order in minimum time.

Table 3: Response time for different number of colonies

Colonies/ Relations	2	4	6	8
10	0.02296	0.02164	0.0228	0.02284
20	0.02724	0.0264	0.0268	0.0272
30	0.057	0.03768	0.04192	0.0388
40	0.06684	0.07892	0.0712	0.0688
50	0.13404	0.13336	0.13444	0.13344
60	0.22604	0.2264	0.22636	0.2254
70	0.4136	0.39056	0.40752	0.40768
80	0.7066	0.72292	0.70036	0.71932
90	1.1552	1.14012	1.15256	1.146
100	1.63384	1.62712	1.65152	1.64672
110	2.46164	2.42404	2.42792	2.4578
120	3.59384	3.59856	3.57956	3.59348

4.3 Experiment 3: Number of Iterations

An important parameter for producing an optimal join order in MCAA-QODD is the number of iterations. In this experiment, MCAA-QODD is executed for a different number of iterations. The number of iterations that were taken into consideration for the experiment is 10, 20 30 and 40. The significance of increasing number of iterations is to achieve the minimum cost as it gives the optimal join order in minimum time. After conducting this experiment, it was observed that the twenty iterations give the optimal join order in minimum Response time.

The second experiment is carried out to find the feasible number of iterations for MCAA-QODD. MCAA-QODD is executed with Different number of iterations i.e. 10,20,30 and 40 iterations. Each iteration number is then executed for 25 times with the number of relations from 10,20,30,………120. The cost and Average Response time are then calculated to check the performance of each option. The Average Response time for each iteration number and the number of relations is given in Table 4.

By considering Table 4, it can be analyzed that the Response time is minimum for ten iterations but after considering the cost associated with each iteration number (Table 5), it was realized that the cost for ten iterations is more than the value achieved from 20 iterations. At few places, cost achieved from 30 iterations is also less but the occurrence of this situation is very less. So after analyzing Table 5, the number of iterations is selected as 20 for MCAA-QODD.

Table 4 : Response Time for different number of iterations

Iteration/ Relations	10	20	30	40
10	0.0232	0.02404	0.02484	0.0236
20	0.02004	0.0264	0.0288	0.02692
30	0.038	0.06432	0.05156	0.05388
40	0.046	0.07892	0.09316	0.11276
50	0.0804	0.1324	0.15916	0.2438
60	0.12724	0.2264	0.3216	0.43064
70	0.25216	0.40672	0.58348	0.78412
80	0.35904	0.72292	1.05224	1.38828
90	0.59376	1.13936	1.69328	2.25044
100	0.82888	1.62712	2.51956	3.26184
110	1.37556	2.43756	3.61168	5.07584
120	1.80444	3.59856	6.124	7.1536

Table 5: Cost for different number of iterations

Iteration/ Relation	10	20	30	40
10	386	266	238.5	278
20	623	513.5	543	562
30	1183.5	836	954	880.5
40	1526.5	1478	1654.5	1517.5
50	1987	1825.5	1626	1743.5
60	2258.5	2214.5	2398	2149
70	2794.5	2718	2719.5	2631.5
80	3425	3009	3062	3028
90	3946	3452.5	3754	3535.5
100	4069	3578.5	3967	4004.5
110	4947	4329.5	4522	4449.5
120	4957.5	4630.5	4794.5	4991.5

4.4 Experiment 4: Response Time of MCAA-QODD and Parallel ACO

Researchers that applied the variant of ACO, PACO (Parallel ACO) gave a solution to query optimization [21]. In PACO, ants of different colonies communicate between different iterations to produce better results by exchanging pheromone values in between them. One of the reasons to choose Parallel ACO is the formula used for calculation of probability to choose the next node and for pheromone update. Another reason is that both of these algorithms apply the same fitness function.

The Parallel ACO algorithm was proposed to deal with the local convergence problem of ACO. It executes the query on 57 colonies and exchanges the value of pheromone between colonies of ants to avoid the problem of local convergence. To compare the algorithms, the Response time of MCAA-QODD and PACO is recorded in Distributed Environment with four sites.

After analyzing the Response time taken by MCAA-QODD and Response time taken by PACO, it can be observed that the time taken by MCAA-QODD is less as compared to PACO. The Difference Percentage indicates that MCAA-QODD provides join order in around 15 percent to 27 percent less time for relations 4, 8, 12, 16……52 less time than PACO. (Table 6) By analyzing Figure 2, it is clear that MCAA-QODD is generating results in less Response Time as compared to PACO. So MCAA-QODD can be considered as a better solution than PACO for join query optimization.

Table 6: Response Time of MCAA-QODD and PACO

Number of Relations	Parallel ACO	MCAA-QODD	Difference	Difference Percentage
4	0.0204	0.0172	0.0032	15.69
8	0.026	0.022	0.004	15.38
12	0.0288	0.02404	0.00476	16.53
16	0.0308	0.02562	0.00518	16.82
20	0.032	0.0264	0.0056	17.5
24	0.0344	0.0272	0.0072	20.93
28	0.03632	0.02764	0.00868	23.9
32	0.08468	0.06432	0.02036	24.04
36	0.08772	0.06692	0.0208	23.71
40	0.10396	0.07892	0.02504	24.09
44	0.142	0.106917	0.03508	24.7
48	0.15364	0.11448	0.03916	25.49
52	0.19076	0.13976	0.051	26.74

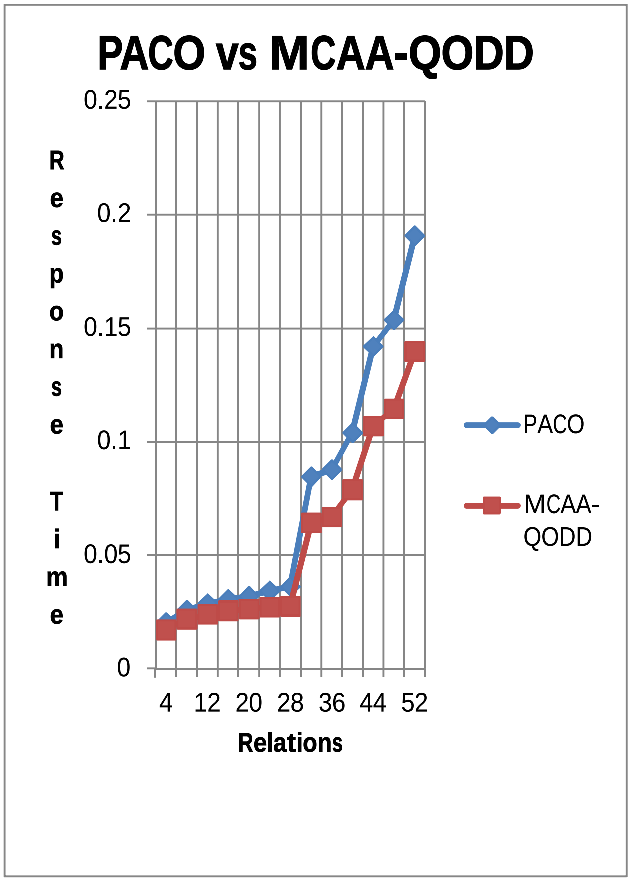

Figure 2: Response Time of MCAA-QODD than PACO

5.CONCLUSION AND FUTURE WORK

As explained above, an improved Optimization algorithm MCAA-QODD performs well for small as well as a large number of relations in a join query. In this paper, the Response Time of MCAA-QODD is compared with another existing algorithm to show the significance of the algorithm. It can be easily observed that the proposed algorithm is reducing around 99 percent Response Time. To validate the effect of parameters on MCAA-QODD, different experiments were conducted. The parameter values are adjusted to receive the optimal solution to the query. The values derived from these experiments help in improving the quality of the result.

MCAA-QODD works well for replicated data. In the future, it can be implemented for fragmented data too. A Modified form of other Swarm Optimization Algorithms such as Particle Swarm Optimization, Artificial Bee Colony, etc. can also be applied to the Join Query Optimization problem.

REFERENCES

[1] A. Kumar and S. Singh. **Improvement of the Performance of a Query Optimization for Distributed System**, *International Journal of Advanced Research in Computer Science and Software Engineering* Vol. 4, Issue 11, pp. 970-975, November 2014.

[2] S. Chaudhuri. **An overview of query optimization in relational systems**,” *in Proc. seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems* **-** *PODS 98*, 1998.
https://doi.org/10.1145/275487.275492

[3] P. G. Selinger, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price. **Access Path Selection in a Relational Database Management System,** *in Proc. 1979 ACM SIGMOD international conference on Management of data*, Boston, 1979, pp. 23-34.
https://doi.org/10.1145/582095.582099

[4] C. Qiao and H. Wang. **The Further Research on the Application of ABC to the Optimization and Control of Project,** *Engineering Management Research*, Vol. 1, no. 2, 2012.
https://doi.org/10.5539/emr.v1n2p96

[5] P. Civicioglu and E. Besdok. **A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms,** *Artificial Intelligence Review*, Vol. 39, no. 4, PP. 315–346, Jun. 2011.
https://doi.org/10.1007/s10462-011-9276-0

[6] N. Li, Y. Liu, Y. Dong, and J. Gu. **Application of Ant Colony Optimization Algorithm to Multi-Join Query Optimization,** *Advances in Computation and Intelligence Lecture Notes in Computer Science*, pp. 189–197, 2008.
https://doi.org/10.1007/978-3-540-92137-0_21

[7] L. Golshanara, S. M. T. Rouhani Rankoohi, and H. Shah-Hosseini. **A multi-colony ant algorithm for optimizing join queries in distributed database systems**, *Knowledge and Information Systems*, Vol. 39, no. 1, pp. 175–206, 2013.

[8] A. Hogenboom, E. Niewenhuijse, F. Hogenboom, and F. Frasincar. **RCQ-ACS: RDF Chain Query Optimization Using an Ant Colony System**, In *Proc. IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, 2012.

[9] A. Soni and S.V. Chande. **Multi Colony Ant Optimization: A New Approach to Query Optimization in Distributed DBMS**, *International Journal of Research in Computer Application and Management*, Vol. 8(10), pp.1–3, 2018.

[10] N. Ahmad, R. H. Salimin, I. Musirin, N. M.S. Honnoon and N Aminuddin. **Solving Economic Load Dispatch for Power Generation Using Genetic Algorithm Techniques**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8(1.3), pp. 337 -344, 2019.
https://doi.org/10.30534/ijatcse/2019/6181.32019

[11] Y. Jhang, P. Agarwal, V. Bhatnagar, S. Balochian, and J. Yan. **Swarm Intelligence and Its Applications**, *The Scientific World Journal* , pp. 43-78, 2013.

[12]D. Hema Latha and P. Premchand. **Estimating Software Reliability Using Ant Colony Optimization Technique with Salesman Problem for Software Process,** *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 7(2), pp. 20-29, 2018.
https://doi.org/10.30534/ijatcse/2018/04722018

[13] S..V. Chande and P. Tiwari. **Optimization of Distributed Database Queries Using Hybrids of Ant Colony**

Optimization Algorithm, *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 3, Issue 6, June 2013.

[14] U. Cekmez, M. Ozsiginan, and O. K. Sahingoz. **Multi colony ant optimization for UAV path planning with obstacle avoidance,** In *Proc. International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016.

[15] M. Mavrovouniotis, S. Yang, and X. Yao. **Multi-colony ant algorithms for the dynamic travelling salesman problem,** *2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, 2014.
https://doi.org/10.1109/CIDUE.2014.7007861

[16] M. Middendorf, F. Reischle and H. Schmeck. **Multi colony ant algorithms,** *Journal of Heuristics*, Vol. 8(3), pp. 305-320, 2002

[17] C. J. Ting and C. H. Chen. **A multiple ant colony optimization algorithm for the capacitated location routing problem,** *International Journal of Production Economics*, Vol. 141, no. 1, pp. 34–44, 2013.
https://doi.org/10.1016/j.ijpe.2012.06.011

[18] R. Vodák, M. Bíland Z. Krivankova. **A modified ant colony optimization algorithm to increase the speed of the road network recovery process after disasters**, *International Journal of Disaster Risk Reduction*, Vol. 31, pp. 1092–1106, 2018.
https://doi.org/10.1016/j.ijdrr.2018.04.004

[19] S. Chowdhury, M. Marufuzzaman, H. Tunc, L. Bian, and W. Bullington. **A modified Ant Colony Optimization algorithm to solve a dynamic traveling salesman problem: A case study with drones for wildlife surveillance,** *Journal of Computational Design and Engineering*, Vol. 6, no. 3, PP. 368–386, 2019.
https://doi.org/10.1016/j.jcde.2018.10.004

[20] A. Aljanaby, K. K. Mahamud and N. M.Norwawi. **Optimizing Large Scale Combinatorial Problems Using Multiple Ant Colonies Algorithm Based on Pheromone Evaluation Technique**, *International Journal of Computer Science and Network Security*, Vol.8 No.10, October 2008.

[21] 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. 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, 2018, pp. 653-656.
https://doi.org/10.1109/ICIVC.2018.8492789

[22] E. Chen and X. Liu. **Multi-Colony Ant Algorithm**, *Ant Colony Optimization - Methods and Applications*, April 2011, pp. 3-12.
https://doi.org/10.5772/13991

[23] S. Alam, R. Barrett, J. Kuehn, P. Roth and J. Vetter. ***Characterization of Scientific Workloads on Systems with Multi-Core Processors***, *2006 IEEE International Symposium on Workload Characterization*, pp. 225-235.
https://doi.org/10.1109/IISWC.2006.302747