Database Query Optimization Based on Parallel Ant Colony Algorithm

Wenbo Zheng[‡†], Xin Jin[¶], Fei Deng[x], Shaocong Mo[§†], Yili Qu[✉††⋆], Yuntao Yang[‖], Xiaojie Li[x], Sijie Long[x], Chengfeng Zheng[x], Jingyi Liu[x] and Zefeng Xie [x]
[‡]School of Software Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[¶]School of Management, Huazhong University of Science and Technology, Wuhan 430074, China
[x]School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China
[§]College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
[††]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China
[‖]School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
e-mail: quwillpower@gmail.com

***Abstract*—Multi-join query optimization is an important technique for designing and implementing database manage system. It is a crucial factor that affects the capability of database. This paper proposes a new algorithm to solve the problem of multi-join query optimization based on parallel ant colony optimization. In this paper, details of the algorithm used to solve multi-join query optimization problem have been interpreted, including how to define heuristic information, how to implement local pheromone update and global pheromone update and how to design state transition rule. After repeated iteration, a reasonable solution is obtained. Compared with genetic algorithm, the simulation result indicates that parallel ant colony optimization is more effective and efficient.**

Keywords-multi-join query optimization; parallel ant colony optimization algorithm; pheromone; heuristic information

I. Introduction

Multi-join query optimization is a difficulty in relational database manage system which has been solved faultily. In traditional applications of relational database manage system, the number of join N involved by a single query is relatively small. Usually, $N < 10$. With the expansion of the database application areas, the traditional query optimization technology cannot support some of the latest database applications. Such as, applications of decision support system (DSS), OLAP and data mining (DM), which may produce a query including more than 100 joins. In this condition, the shortfall of the traditional query optimization technology is exposed gradually. Therefore, it is necessary to explore new technology to solve multi-join query optimizatio problem.

Multi-join query optimization is an NP hard problem [1]. With the increase of join number, the number of query execution plan (QEP) corresponding to a query grows exponentially, which lead to computational complexity of multi-join query optimization problem is very large. Recently, solving the problem with heuristic algorithm becomes a hotspot. Such as, Greedy Algorithm [2], KBZ Algorithm [3],GA [4], ABC [5] etc. Ant Colony Optimization(ACO) as a highly effective optimization algorithm has been successfully applied to some classic compounding optimization problems. But many ACO methods [6]–[10] could accelerate the search, but did not help with ACO's problem of local convergence. Wenbo Zheng [11] proposed the parallel ant colony optimization(PACO) to solve ACO's problem of local convergence. In this paper, PACO was adopted to solve the problem multi-join query optimization.

[†]Wenbo Zheng and Shaocong Mo contribute equally to this study.
[⋆]Yili Qu is the corresponding author.

II. Parallel Ant Colony Optimization

Ants communicate through pheromone in each one of ant colony optimization(ACO)'s iteration. [12] This natural characteristic provides a good basis for implementing parallelism in the algorithm. Therefore, it is possible to transform the serial ant colony algorithm into a parallel one.

The algorithm updates pheromone based on the calculation of a transition probability formula

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[h_{ij}(t)]^{\beta}}{\sum_{s \in allowed_k}[\tau_{is}(t)]^{\alpha}[h_{is}(t)]^{\beta}}, j \in allowed_k \\ 0, otherwise \end{cases} \quad (1)$$

where τ_{ij} is the pheromone on edge (i, j), is the heuristic factor of moving from node i to node j, $allowed_k$is the set of cities accessible for ant k in the immediate instance.

To prevent ants from returning to previously accessed cities, a taboo list $tabu_k$ is used to record all traversed cities of an ant k. After time t has passed, all ants have completed a path. The shortest of all paths are recorded and pheromone is updated on each edge.

Ants will then choose its next node, $j(j \in \{0, 1, \ldots, n-1\} - tabu_k)$, and is relocated there. The accessed node will then be placed into the taboo list.

During iterations, pheromone evaporation is processed first, then the pheromone release. The formula for this process is as follows:

$$\tau_{ij} = (1 - \rho)\tau_{ij} \quad (2)$$

where $\rho(0 < \rho \leq 1)$ is the pheromone evaporation coefficient. Also

978-1-5386-4991-6/18/$31.00 ©2018 IEEE

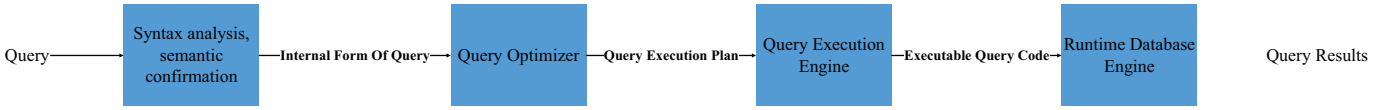

Figure 1. Process of query execution

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{3}$$

where $\Delta\tau_{ij}^{k}$ shows the pheromone released by ant k when it goes through the edge, it is defined as

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{d_{ij}}, edge(i,j) \in T^{k} \\ 0, otherwise \end{cases} \tag{4}$$

Pheromone on every path is updated accordingly. The steps taken to realize parallelism for ACO is shown in Appendix 1. The parallel ant colony algorithm consists of three parts: initialization, cycle, and termination.

Also, when all ants in the colony have found a path they deem optimal, the paths are compared in length for the shortest, which will be regarded as the global optimal solution. The example being parallelization on MIC architecture.

Algorithm 1: Parallel Ant Colony Algorithm

Initialization:

1: Initialize parameters, enter node coordinates and pheromone strength on each path, initialize pheromone matrix and set pheromone increment edges to zero.
2: Set the number of threads, P, according to the number of computational cores ($P = 57$), thereby dividing a colony into 57 smaller colonies for parallel solution.
3: Place ants randomly into cities.

A cycle is as follows:

1: Parallel section begins, and the ***#pragma parallel omp*** for is used to achieve parallelism for the ***for*** loops.
2: Edge pheromone is updated according to the transition probability formula, and calculate the length covered by each ant. The shortest path is selected.
3: Parallel section ends, pheromone is updated.
4: Empty all ants taboo lists.

Termination condition:

The algorithm is terminated when at least one of the following is true:

1: The optimal solution is found.
2: Number of cycles reaches threshold.
3: Search time reaches threshold.
4: No better global solution is found after a certain number of cycles are completed.

III. Proposed Query Optimization Algorithm

The process of relational database manage system managing user query is as follows: After receiving query submitted by users, query parser checks syntax, verifies relations, translates the query into its internal form. It is usually translated into relational algebra expression, which can be denoted as query syntax tree. A relational algebra expression may have many equivalent expressions, so it also corresponds to many equivalent query syntax tree. Then, query optimizer selects appropriate physical method to implement each relational algebra operation and finally generate query execution plan (QEP). The QEP consists of the order in which the operations in a query are to be processed, and the physical method to be used to process each operation. Amongst all equivalent QEP, query optimizer chooses the one with lowest cost output to the query-execution engine, Then, the query-execution engine takes the QEP, executes that plan, and returns the answers to user. The process is depicted in Fig. 1. This paper is to study how to make query optimizer select a QEP with lower cost in shorter time.

Algorithm 2: Database Query Optimization based on Parallel Ant Colony Algorithm

Input: Collection of all relations corresponding to query Q called L.

Output: Join tree M with special join order making query execution cost lowest

1: **for** $i = 1, 2, 3, ..., m$ **do**
  for $j = 1, 2, 3, ..., n$ **do**
    $M = R$
    while L *is not null* **do**
      if *relation* C *and current relation node are linked with edge in query graph* **then**
        According to state transition rule choose next relation node C
        $M = M \cup C$
        Update query graph, delete edge $E(R_i, C)$, R_i is current node
        R_i and C are merged into as current relation node
        $L = L - C$
        Local update pheromone
    $n = n + 1$
  Record the cost of optimal solution $Q(m)$ in this iteration
  Global update pheromone
  $m = m + 1$
2: Record global optimal solution M

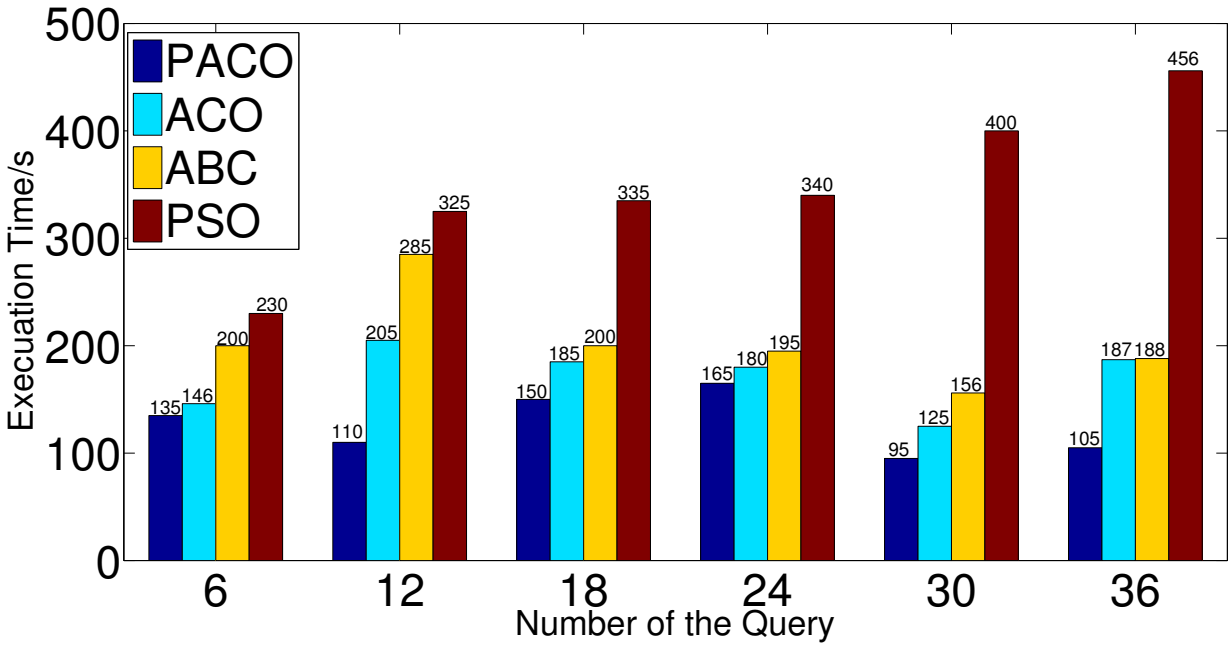

Figure 2. Effect of number of queries based on 36

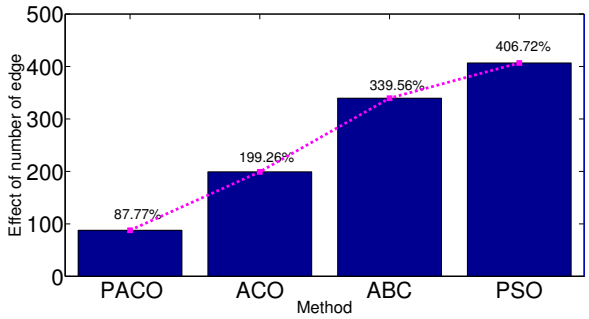

Figure 3. Effect of number of edge

A. Mathematical Model of Database Query Optimization

For a connection $J = R\ join\ S$, the cost function of the J is

$$cost(J) = \frac{|R| \times |S|}{\prod_{c_i \in C} \max(V(c_i, R), V(c_i, S))} \tag{5}$$

where the function v is

$$V(c, J) = \begin{cases} V(c, J), c \in R - S \\ V(c, R), c \in S - R \\ \min(V(c, R), V(c, S)), c \in R \cap C \\ 1 \leqslant i_1, i_2, \ldots, i_m < n \end{cases} \tag{6}$$

B. Query Optimization Algorithm

Query optimization based on parallel ant colony algorithm is proposed in which a set of artificial ants builds feasible solutions with some join order. Because of positive feedback mechanism, the ant could collaborate with each other and could find the better solution in shorter time. Our works as shown in Algorithm.2.

IV. Experimental Evaluation

In order to explain the effect of bees in solving the multi-join query optimization problem, all experiments were conducted using a compute node in Tianhe-2 with two Xeon E5 12-core CPUs, three Xeon Phi 57-core coprocessors, 88GB of RAM, and Ubuntu Linux. However, all experiments need a compute node with 4-core CPU, a Xeon Phi 57-core coprocessor, 16GB of RAM, and Ubuntu Linux in practice, which generates a database of 52 relations where each relation cardinality is in $[10, 110]$ and 36 Query. The experiment execution in Java NetBeans with MySQL server 2014 adventure work database 2014. The query categorized into ten sets of queries of different size (i.e. number of query is $6, 12, 18, 24, 30, 36$).

In this section, we quantify the optimization overhead of our approach and show that the overhead incurs only a very small fraction of the total query processing time. Since the optimization times for our algorithms do not show much differences. In this section, we present an experimental study to evaluate our proposed approach. Effect of number of queries. Compares the performance as the size of a query batch is increased. Observe that our algorithms significantly outperform NA in Fig. 2. For example, our outperforms PSO by 77% on average and up to 106% when the number of queries is 30. Effect of number of edge, when we have 36 query there are 36 edge our algorithm reduce query execution plane to 14 as Fig. 3.

Queries in this section, we study the efficiency of our proposed for multi-join query optimization. To generate a batch of queries, we first generated$N$ (the default value is 10) relations. As discussed previously, we then generated the

cardinalities for each relation as well as the selectivity factors for each pair of relations representing a join predicate between them. Finally, each query in a batch was generated as follows:

1) we first randomly chose a subset of the N relations for the query and then generated a random acyclic query for the chosen relations.
2) we chose random acyclic queries since they are more common in real life applications.For example, when $N = 10$ and the number of queries in a batch is 20, it took only 32 s to optimize the queries.

After analysing the results of an experiment this can be concluded that the proposed approach in this paper is more effective and efficient than the benchmarked PSO [13] solution. The proposed approach provides an optimal solution which is faster than the PSO [13], ACO [1], ABC [14] and also gives better quality of solution.

V. Conclusion

Multi-join query optimization problem is hotspot in database research field. A good optimization algorithm not only can improve the efficiency of queries but also reduce query execution costs. In this paper parallel ACO algorithm was proposed to solve the problem of multi-join query optimization. In our method, local pheromone update and global pheromone update are applied, and at last obtain a rational solution. The simulation results show that parallel ACO finds optimum solutions more effectively both in time and quantity than other methods which are swarm intelligence algorithm especially with the increment of relation number.

Acknowledgment

This paper was supported in part by Science & Technology Pillar Program of Hubei Province under Grant(#2014BAA146), Nature Science Foundation of Hubei Province under Grant (#2015CFA059), Science and Technology Open Cooperation Program of Henan Province under Grant (#152106000048) and Fundamental Research Funds for the Central Universities (#2018-JSJ-B1-07, #2018-JSJ-A1-02, #2018-JSJ-A1-01, #2018-JSJ-B1-05, #2018-JSJ-B1-06, #2018-JSJ-B1-08, #2018-JSJ-B1-12, WUT:2017II03XZ).

References

[1] N. Li, Y. Liu, Y. Dong, and J. Gu, "Application of ant colony optimization algorithm to multi-join query optimization," vol. 5370, pp. 189–197, 2008.

[2] Y. Cao, Q. Fang, G. R. Wang, and G. Yu, "Parallel query optimization techniques for multi-join expressions based on genetic algorithm," *Journal of Software*, vol. 13, no. 2, pp. 250–257, 2002.

[3] A. N. Swami and B. R. Iyer, "A polynomial time algorithm for optimizing join queries," in *International Conference on Data Engineering, 1993. Proceedings*, pp. 345–354, 2002.

[4] V. Tereshko and A. Loengarov, "Collective decision-making in honey bee foraging dynamics," *Computing and Information Systems Journal, ISSN 1352-9404*, 2005.

[5] M. Alamery, A. Faraahi, H. H. S. Javadi, S. Nourossana, and H. Erfani, "Multi-join query optimization using the bees algorithm," in *Distributed Computing and Artificial Intelligence - International Symposium, Dcai 2010, 7-10 September 2010, Valencia, Spain*, pp. 449–457, 2010.

[6] R. Jovanovic and M. Tuba, "Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem," *Computer Science and Information Systems*, vol. 10, no. 1, pp. 133–149, 2013.

[7] S. Gajjar, M. Sarkar, and K. Dasgupta, "Famacrow: Fuzzy and ant colony optimization based combined mac, routing, and unequal clustering cross-layer protocol for wireless sensor networks," *Applied Soft Computing*, vol. 43, pp. 235–247, 2016.

[8] K. JIANG, M. LI, and H. ZHANG, "Improved ant colony algorithm for travelling salesman problem," *Journal of Computer Applications*, vol. S2, pp. 114–117, 2015.

[9] Q. Cai, D. Zhang, W. Zheng, and S. C. Leung, "A new fuzzy time series forecasting model combined with ant colony optimization and auto-regression," *Knowledge-Based Systems*, vol. 74, pp. 61–68, 2015.

[10] N. Jalil, S. Julai, and R. Ramli, "Parametric modelling of flexible plate structures using continuous ant colony optimization," *Journal of Simulation*, vol. 9, no. 3, pp. 223–231, 2015.

[11] W. Zheng, S. Mo, Y. Qu, X. Jin, J. Zhou, P. Duan, and T. Zheng, "Pattern learning based parallel ant colony optimization," in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications*, pp. 497–502, 2017.

[12] A. Rashno, B. Nazari, S. Sadri, and M. Saraee, "Effective pixel classification of mars images based on ant colony optimization feature selection and extreme learning machine," *Neurocomputing*, vol. 226, pp. 66 – 79, 2017.

[13] R. I. Chang, S. Y. Lin, and Y. Hung, "Particle swarm optimization with query-based learning for multi-objective power contract problem," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3116–3126, 2012.

[14] 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," 2015.