A Review of Algorithms for the Join Ordering Problem in Relational Database Systems

[1]Chintal Upendra Raval, [2]Professor Kaushal Madhu
[1] Student of Gujarat Technological University, [2] Assistant Professor at L. J. Institute Of Engineering & Technology
[1]Department of Information Technology,
[1]L. J. Institute of Engineering and Technology, Gujarat Technological University, Ahmedabad, India

***Abstract* - Finding the optimal join ordering for a database query is a complex combinatorial optimization problem which has been approached by a wide variety of strategies and algorithms, ranging from simple deterministic search to complex hybrid algorithms based on genetic search and incorporating domain-specific heuristics. In this paper we review a set of join ordering algorithms and classify them according to the nature of the search strategy they implement. We also briefly discuss the relative advantages and applicability of different algorithms.**

***Index Terms* - Query optimization; join ordering, relational databases: query execution plan; deterministic algorithms; randomized algorithms; genetic algorithms; hybrid algorithms, Multi Join Query Ordering.**

I. INTRODUCTION

Query Optimization is one of the most important and expensive stages in executing database queries. If these queries involve join operator between several tables, join ordering process will have a considerable effect on lowering costs of execution. This operator relates two tables through their common attributes. It is the responsibility of optimizer to find an execution plan with minimum costs or almost close on that. In queries with maximum 5 or 6 relations, the best order is easily reachable using evaluation and search in the whole possible space, and this can be accomplished in shorter than a second. But, in the case of more than 8 relations, it is not possible to find the best plan easily [4].

In traditional relational database, the number of relations in join queries is usually less than 10. In these cases, such methods as dynamic programming have been used in order to cope with this difficulty. However, in such systems as decision backup systems, data mining and OLAP, sometimes, there are more than 100 tables in join operator [4].

A central issue in relational query optimization is the selection of an effective join ordering, i.e., an order for evaluating efficiently the join predicates of a given query. For example, when joining 3 tables A, B, C of size 10 rows, 10,000 rows, and 1,000,000 rows, respectively, a query plan that joins B and C first can take several orders-of-magnitude more time to execute than one that joins A and C first [5].

The number of execution plans, among which only the most appropriate one would be adopted, increases as The size of data gets bigger and the number of relations participating in join operator arises, but traditional methods do not give proper answer for this problem [4].This problem is generally considered as NP-Hard problem [4].

The Join Ordering Problem (JOP) has been approached by several classes of algorithms. It is a generalization of the classical combinatorial Traveling Salesman Problem (TSP). The problem of finding the shortest Hamiltonian cycle in a complete graph. The TSP is among the best-studied comma trial optimization problems and dozens of algorithms have been proposed for it. Most of these algorithms are directly applicable to the JOP (which is considerable newer). In this review however, we consider only algorithms already applied to the JOP. An extensive survey of general global optimization algorithms is not in the scope of this work.

II. DESCRIPTION OF THE JOIN ORDERING

A. Join Ordering Problem

When query command entered by user, at first, Parser analyses the commands syntactically, and transforms it into a standard form for next time if no error has been recognized. Next, optimizer receives the standard form and finds an execution plan, and transmits it to query execution engine. Finally, when execution of codes completed, the query results will be returned. **"Figure 1"** demonstrates these stages [4].

In optimization problem, input is a query graph (join graph) including all participant relations (tables) in join. These relations are considered as graph nodes. Search space or solution space is a set of plans providing same results for the problem. A solution is described by processing tree showing examination of join statements. Processing tree is a binary tree which its leaves and internal nodes are basic tables and join operator, respectively. Edges determine current of data movement from vertices to root [4].

Although finding an optimum execution plan for query is theoretically possible, in most cases, optimizers provide one effective and acceptable execution plan [4].

In addition to the join ordering, type of join operation is another parameter having considerable effect on costs of final execution. Join can be of Nested Loop Join, Merge Join, or Hash Join type. In the present study, Nested Loop Join is the only type to be assessed.

The goal of optimization is to find a point with the minimum cost in search space. Cost can be measured as tuples which should be read from disc and/or written on disc. In this study, it is assumed that execution environment is not distributed and that the content of database is much bigger than the capacity of main memory [4].

Five relations, known as R1, R2, R3, R4, and R5 have been taken into account, which participate in a query command Q in join operator. Processing tree can be left-deep, right-deep or bushy type. In left-deep trees, right part of each node is consistently a basic relation. Generally, with n relations, it may create $\binom{2(n-1)}{n-1}(n-1)!$ processing tree and n! Left-deep tree. **"Figure 2"** shows three kinds of join tree [4].

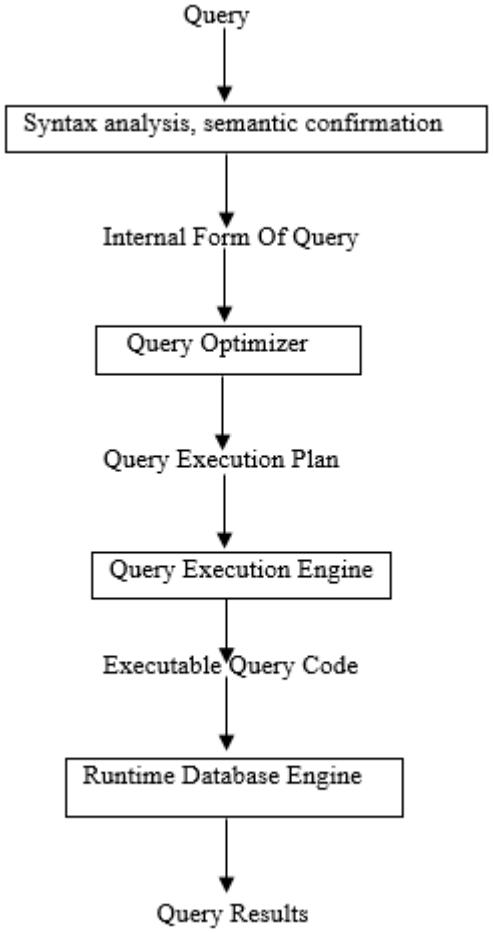


Figure 1: Process of query execution [4].

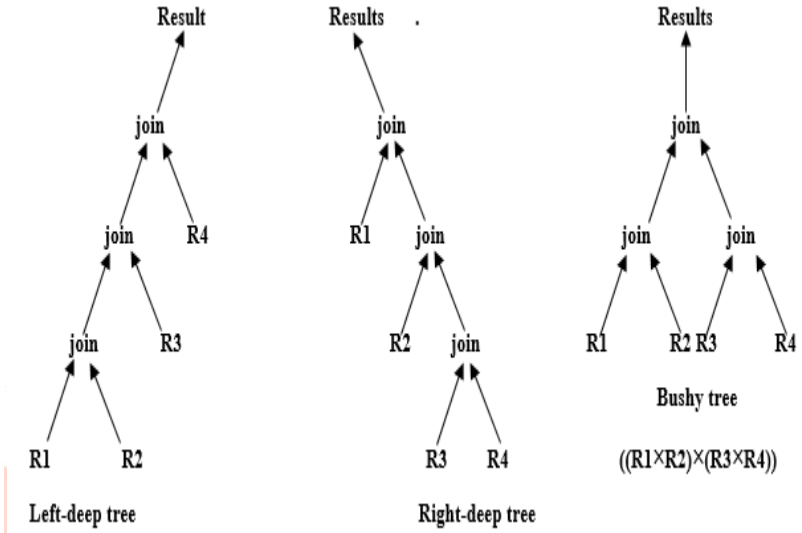


Figure 2: Three Kinds of Join Tree [4]

B. Cost Function

To estimate the cost of join process between several relations, a simple model is used: the sum of the tuples number about intermediate results decides the cost of QEP. The required parameters are:

N(R): Number of Tuples in Table R.

D (A, R): Numbers of Distinct Values of Attribute A in Table R.

It is assumed that the values of fields in the tables are distributed evenly. Considering two relations R1 and R2 and their common attributes X, the number of tuples resulting from their join process is

$$n(T) = \frac{n(R1)\times n(R2)}{\max(D(X,R1),D(X,R2))} \qquad (1)$$

III. JOIN ORDERING ALGORITHMS

A. Genetic Algorithm

Designed to mimic the natural evolution process, genetic algorithms nowadays enjoy an increasing popularity and are being applied to various complex optimization problems. As in Nature, where the best fit individuals in a population have greatest survival probability and highest opportunity to have their features inherited by the offspring, genetic algorithms breed and combine solutions to obtain even better ones.

The Query execution plan can be considered a program in an abstract tree representation which is evaluated bottom up. The relations are the terminals and the joins are the functions in the genetic program. Thus, the Query execution plan satisfies the structural requirements of the genetic programming method, which applies the paradigm of search through genetic algorithms. The input and the output of each join operator in the plan are relations, therefore the closure requirement defined in is satisfied [6].

Each particular genetic algorithms is a concretization of the canonical genetic optimization algorithm, that is, different genetic algorithms differ from each other in their coding method (converting a solution into an internal representation upon which the genetic operators can be applied) and by the choice of the three genetic operators - selection, mutation and crossover[6].

Genetic algorithms have been first applied to the JOP, the fitness function used requires backward transformation from chromosome to tree representation, which is complex and with high computational cost. The chosen cross- over operators have a serious flaw - they disrupt the chromo- some structure, transforming two valid parent chromosomes into an invalid one, which then needs to be "repaired" to become a correct solution encoding. Despite these shortcomings, the achieved results are promising [6].

Later in some of these disadvantages have been overcome. The fitness function is based on the cost of the query execution plan, which is defined as the total execution time from the first retrieval of a relation from the database to the completion of the output generation (the Query result). The model also considers multiprocessor environment and implements parallel processing [6].

Currently, probably the most popular non-experimental genetic Sal Query optimizer is the GEOO (Genetic Query Optimizer) in the PostgreSQL RDBMS. It considers only left-deep solutions, implements an Elitist selection operator, a simple edge recombination crossover and does not apply mutation. The population size is fixed. PostgreSQL has two optimizer implementations, a classical deterministic optimizer and a genetic optimizer, the latter being used for Queries with more than 10 joins. A genetic query optimizer was also introduced in the Microsoft SOL Server 2005 [6].

a. Coding

The coding methods for the JOP can be classified by the form of the trees they operate on - left-deep or bushy the choice of coding strongly influences the choice of the three genetic operators. Left-deep coding are prevalent with genetic algorithms. Mostly because they allow for simpler and more efficient mutation and crossover implementations. Note that the search space must be closed under the genetic operators which means mutation and crossover should produce valid solutions with respect to the selected coding.

Each left-deep tree can be represented in a unique way as an ordered sequence of its leaves:

$$((((R1 \infty R5) \infty R3) \infty R4) \infty R2) \rightarrow 15342$$

This is probably the most popular coding - it has been used in numerous genetic algorithm implementations including PostgreSQL's GEOO.

i. Simple Left-Deep Tree Coding

An ordered list of all relations participating in the join is created. The solution is scanned from left to right, each relation is substituted by its index in the ordered list and then it is removed from the list. Such a coding has been successfully used in the solving of the TSP with genetic algorithms [6].

ii. Travelling Salesman Coding

The coding of bushy trees is less straightforward. It must be designed so as not to overly complicate the implementation of the crossover and mutation operators. A good coding algorithm has been proposed, where the symbols in the code represent the edges of the query graph. One useful property of this coding is that it cannot represent Cartesian products, which means that the application of any crossover and mutation operators cannot lead to a Cartesian product in the execution plan [6].

$$L\,[1, 2, 3, 4, 5]\ ((((R1 \infty R5) \infty R3) \infty R4) \infty R2) \rightarrow 14221$$

b. Selection

i. Roulette Selection

Each individual in the population corresponds to a disc sector whose area is inverse- proportional to its cost. The disc can be thought of as a roulette. Turns of the roulette determine the N individuals of the new generation. This algorithm considers the relative fitness of the individuals in the population, which means that a "super" individual may cause the early extinction of other individuals. The classical Roulette selection is thus characterized by fast convergence [6].

ii. Magnitude Roulette Selection

This is a variation of the classical Roulette selection, in which the disc areas are determined not by the fitness of the individuals, but by the magnitude of their fitness. Experiments show that the Magnitude Roulette selection is characterized by slower evolution progress but the risk of premature convergence is much lower [6].

iii. Rank Selection

In the domain of Query optimization. The fitness of the individuals in a population may vary by 10100• in selection algorithms based on the relative fitness of the individuals, some will have no chances to survive while others will quickly dominate the population, leading to quick convergence. The rank selection algorithm assigns ranks from N for the best-fit Individual to 1 for the worst fits individual in a population with size N. Then each individual with rank R survives with probability R / ((N + 1) * N /2) [6].

iv. Elitist Selection

The individuals in the population are sorted in decreasing order of their fitness and the first N individuals are preserved in the new generation. This is the most popular selection operator and it is characterized by relatively fast convergence. It is the selection algorithm implemented in GEOO [6].

v. Adaptive Selection

Self-adaptation in genetic algorithms (population size ad- adaption in particular) is a topic that is receiving considerable attention recently. The classical selection algorithms keep the population size fixed. This simplifies the algorithms but it is an artificial restriction and does not follow any analogy to biological evolution, where the number of individuals in a population varies continuously in time, increasing when there are high fit individuals and abundant resources and decreasing otherwise. Intuition hints that it may be beneficial for the population to expand in the early generations when there is high phenotype diversity and there is opportunity to "experiment" with different characteristics of the individuals, and to shrink with the increase of population convergence. When the unification of the individuals in terms of structure and fitness no longer justifies the maintenance of a large population and the higher computational costs associated with it. An adaptive selection operator with dynamic population size has been recently applied to the JOP and its comparison against the classical fixed-size Elitist selection seem promising [6].

c. Crossover

i. Subsequence Exchange Crossover I

This crossover algorithm is applicable to the Simple Left- Deep Tree Coding and the Bushy Tree coding. A random subsequence of the code characters of both parents is chosen. Then the subsequence is substituted by another one containing the same characters but arranged in the order of their occurrence in the other parent [6].

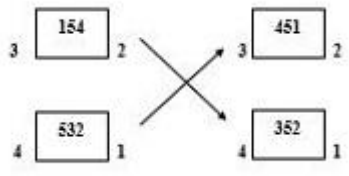


Figure 3: Subsequence Exchange Crossover I [6].

ii. Subsequence Exchange Crossover II

This crossover algorithm IS applicable to the Traveling Salesman coding. Two random subsequences with equal length are chosen in both parents. Then the two subsequences are exchanged to form the offspring. This coding is applicable only to the Traveling Salesman coding because it allows duplicate symbols in the code [6].

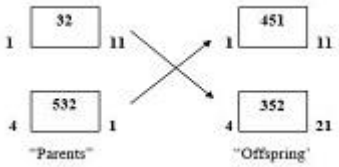


Figure 4: Subsequence Exchange Crossover II [6].

iii. Subsets Exchange Crossover

This crossover algorithm is applicable to the Simple Left Deep Tree Coding and the Bushy Tree Coding. Two random subsets of characters with equal cardinality are chosen in the codes of both parents such that the two subsets contain the same elements. Then these subsets are exchanged to form the offspring [6].

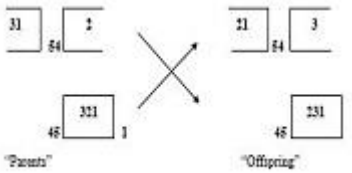


Figure 5: Subsets Exchange Crossover [6].

iv. Order Crossover

Offspring are generated by choosing two random split points in the parent chromosomes, inhering a gene subsequence from one of the parents and filling up the missing genes in the relative order they occur in the second parent [6].

(1 3 | 5 7 9 10 | 2 8 6 4) → (7 10 | 2 1 6 9 | 8 4 3 5)
(3 8 | 2 1 6 9 | 8 4 3 5) → (1 6 |5 7 9 10 | 4 3 8 2)

v. Modified Two Swap (M2S) Crossover

This is one of the two crossover operators proposed in tile first work to apply genetic optimization to the JOP. Two genes are randomly chosen in the first parent and are replaced by the corresponding genes from the second parent preserving their order III the second parent [6].

(1 3 2 4 6 5) → (4 3 2 1 6 5)
(2 3 4 6 5 1) → (2 1 4 6 5 3)

vi. CHUNK Crossover

This is the second crossover operator for bushy encoding. A random chunk of genes in the first parent is chosen, the chunk is copied into the offspring (in the same position it occurs in the parent) and the rest of the genes in the offspring are filled up in the order they occur in the second parent [6].

d. Mutation

i. Reciprocal Exchange Mutation

The genes in (random) position i and (i + 1) mod N, where N is the length of the chromosome, are swapped and applied to the simple left-deep tree coding. This mutation operator obtains a new chromosome that is a valid solution [6].

ii. Exhaustive Mutation

With left-deep coding where a gene contains information about a relation and a join method, the following simple mutation can be considered: two random genes i and j are swapped and the join method of another randomly selected gene k is modified. Such a mutation operator, even if applied alone, guarantees that every point in the solution space is reachable for any choice of the starting point. This is a useful characteristic of genetic operators since it is a prerequisite of the convergence of the genetic algorithm [6].

B. Combination Of Genetic Algorithm And Ant Colony

a. Ant Colony Algorithm

Ant colony algorithm has been introduced by Marco Dorito in 1992, and it has been yet used to solve many problems dealing with optimization. He has inspired from real ants and routing between food and their nest. Firstly, each ant has been placed to the same relation R which Number of tuples about R is smallest. After that, ants begin to move. Each ant uses transition rule in order to select the next relation (for join process), and after it passes a route, they leave their tracks. This is made by pheromone. In order to reduce the search space furthermore, avoiding the emergence of Cartesian product often is considered as constraint of the issue. Transition rule: In the case an ant placing on relation I wants to select the next relation (j) for join process, it applies transition rule and considers those relations which do not produce Cartesian product.

Transition rule for each ant defined as:

$$v = \begin{cases} argmax\left[(\tau_{ij}^{\alpha})(\tau_{ij}^{\beta})\right], q < q_0 \\ V, q > q_0 \end{cases} \quad (2)$$

Where τ_{ij} is the amount of pheromone on the edge (i, j) and $^{\eta}{}_{ij}$ is reversion cost for joining between the relations i and j, respectively. Transition rule is based on probabilities. Parameters _ and _ are used to adjust the effect of parameters 7 and8 .q is produced randomly between (0, 1). When q is less than threshold level q0, the relation (j) producing the highest value in $(\tau_{ij}^{\alpha})(\tau_{ij}^{\beta})$ is selected as the next destination for ant; otherwise, the next relation is determined through following equation:

$$p_{ij} = \begin{cases} \frac{(\tau_{ij}^{\alpha})(^{\eta}{}_{ij}^{\beta})}{\sum_{j \in Remain}(\tau_{ij}^{\alpha})(^{\eta}{}_{ij}^{\beta})} & j \in Remain \\ 0 & otherwise \end{cases} \quad (3)$$

Remain is a set of relations which are not selected yet, and also, selecting them do not make Cartesian product.

Updating pheromone: In ant colony algorithm, updating pheromone is conducted in two stages: Local update: When passed each route (joining process), ants leave some pheromone determined by the following formula:

$$\tau_{ij}^{new} = \rho \times \tau_{ij}^{old} + (1 - \rho) \times \tau_0 \quad (4)$$

Where τ_0 the initial value of pheromone on graph edges, defined as is $\tau_0 = 1/N \times C_{min}$ is the numbers of relations participating in join process, and Cmin denotes the cost resulting from applying greedy algorithm on join tree.

i. *Global update*

When ants complete the tour in each stage, global update is performed. This rule is only applied on the route with the least cost. In fact, doing so makes probability of this route to be selected by the following ants becomes higher.

$$\tau_{ij}^{new} = \rho \times \tau_{ij}^{old} + (1 - \rho) \times \Delta\tau_{ij} \quad (5)$$

Where $\tau_{ij} = 1/Q(M)$and Q (M) is cost of the best route to be found in each algorithm iteration [4].

b. Genetic Algorithm and Ant Colony Algorithm: A Combination

Combined methods for optimization have a considerable effect on high convergence speed and low execution time.

Statistically, genetic algorithm is of a higher population scattering than ant colony algorithm and has a capability to search extensive amplitude of answers. In this algorithm, convergence speed toward optimum answer is lower than ant colony algorithm. In contrary, ant colony algorithm has a low population scattering and high convergence speed. In this study, a combined method, based on genetic algorithm and ant colony algorithm, has been implemented.

In this way, at first, a set of execution plans is established by artificial ants, and then, these plans will be considered as initial population of genetic algorithm. Next, pheromone in routes will be answers passed into genetic section, then, based on reproduction resulting from their pair in genetic algorithm, a new generation is developed and the answers passed to ant colony. Then previous process is iterated. The experimental results show that convergence rate of produced offspring in each generation, on average, is better than the preceding generation of basic genetic algorithm, and also a closer answer to optimum updated.

In each iteration, each ants, firstly seek answers, these one would be produced comparing to ant colony algorithm. “Figure 3” shows the order of above stages. That is, in each generation, in addition to high dissipation (in order to bypass local minimums), convergence speed toward optimum answer becomes higher than previous methods.

Modeling genetic algorithm for join ordering problems: Genetic algorithm can easily model join ordering problem. Every chromosome is a solution for the problem. Each gene in chromosomes denotes a table in relative query. Each chromosome Ci is made up of a number of genes Gj. Each gene is a table for a query. In one generation, depending on its relative population size Pk, number of chromosomes is variable. “**Figure 4**” shows structure of a chromosome.

Selection operator: it takes population of one generation and selects a number of them to transfer to the next generation.

Mutation operator: it takes a chromosome as input and produces a new chromosome.

To select a number of chromosomes for transferring to the next generation, the quality of chromosomes (determined by cost function) will be considered. The reverse of total runtime for executing queries is a simple selection for evaluation function. Moreover, crossover and mutation operators can be easily defined for join ordering problem. These operators cause new and valid solutions for the problem. As genes in a chromosome express a table for query related to that gene, so replacing existing table with another one

makes it to establish a new and valid solution. Simply it produces two random numbers and exchanges the related genes. This is performed by mutation operator [4].

Crossover: The crossover operator combines the chromosomes of two generation individuals-g C_i^g and C_j^g to obtain two new generation-(g + 1) individuals C_i^{g+1} and C_j^{g+1}. Random locus x is chosen, the two parent chromosomes are split at that locus and each of the two offspring receives a whole fragment from one of the parents (the first child - the left and the second child - the right relative to the locus) and the rest of the chromosome is filled up with the missing genes in the order they occur in the second parent. This guarantees both structural and functional similarity of the children with both their parents [4].

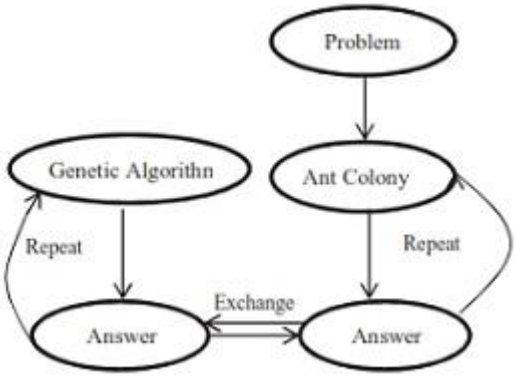


Figure 3: Structure of Algorithm [4].

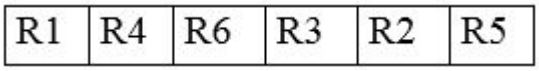


Figure 4: Structure of Chromosome [4].

C. Cuckoo and Tabu Search

a. Cuckoo Search

The Cuckoo search algorithm, which is a population based scholastic global search algorithm, discovered by Yang and Deb is inspired by breeding strategy of cuckoo species. This algorithm is based on following three assumptions;

1. Each cuckoo lays one egg at a time, and chooses a nest randomly to dump this egg;
2. The best nests with high quality of eggs (solutions) will carry over to the next generations.

The number of available host nests is fixed, and a host can discover an egg dumped by cuckoo with a probability pa ∈ [0, 1]. Then host bird can either throw the discovered egg away or abandon that nest so as to build a completely new nest.

For simplicity, last assumption can be approximated by a fraction pa of the n nests being replaced by new nests (with new random solutions at new locations). Levy fly is preferred to generate the new solution (i.e. cuckoo egg) for better results. Levy fly provides a way to perform a random walk, where random step length is drowning from levy distribution. Levy flight is inspired by flight characteristic behavior of many birds and insects, leading to scale free search pattern [3].

b. Tabu Search

Tabu search, developed by Fred Glover in 1970, is the search strategy where memory and search history is its integrated part. It is an intensive local search algorithm, which uses the memory (search history) to avoid the potential cycling of local optimal solutions. A Tabu list is maintained, which contains all recently visited solutions to avoid moving again and again to these solutions. This strategy saves time by avoiding the repetition of previous moves and prevents it from being stuck at local minima.

MJQO is an optimization problem which is modeled as a search algorithm where the optimal QEP has been searched from all post- sable QEPs. Heuristic techniques are effective techniques to solve such kind of applications. Cuckoo Search technique is superior to all other existing heuristic techniques. Proposed solution for MJQO problem uses cuckoo search algorithm to find optimal QEP to improve the result of previous existing solutions of this problem. Cuckoo search improves the running time. In some cases cuckoo search algorithm gets stuck at local minima and is not able to give best possible optimal solution. Proposed approach uses Tabu search to identify such situations and levy flight to come out of this situation. It improves the quality of solution. As mentioned above, both parameters, execution speed as well as quality of solution are equally important in MJQO and should be improved. Proposed solution addresses both the parameters.

c. Cuckoo and Tabu Search Combined Algorithm

Cuckoo search is based on the breeding strategy of cuckoo species. In this approach initially some eggs are there. After validating the eggs' quality some eggs are abandoned and some new eggs will replace these abandoned eggs whose quality is better than those abandoned eggs. By repeating these steps final outcome will be the best quality of eggs. In proposed algorithm each possible QEP acts as an egg. Initially some random QEP (initial cuckoo eggs) are generated and then follows cuckoo's breeding behavior. It will generate new QEP (i.e. similar to laying a new egg) and compare the newly generated QEP with any randomly selected QEP from initial set. If the quality of newly generated QEP is better to randomly selected QEP then randomly selected QEP will be abandoned and newly generated will replace it. If we repeat this cuckoo behavior after some iterations we will get a set of best QEP among all possible QEPs of input query [3].

According to algorithm when a query graph comes to query optimizer, it will randomly generate the set of initial QEPs (i.e. cuckoo eggs) called S and after it, a new QEP called e will be generated using a generator function. Permutation function P1, used as generator function. P1 is a permutation function which generates different permutations of relations in a query graph by reordering the position of relations in the query graph. Main concern in newly generated QEP by P1 is that it should fulfill all the constraints

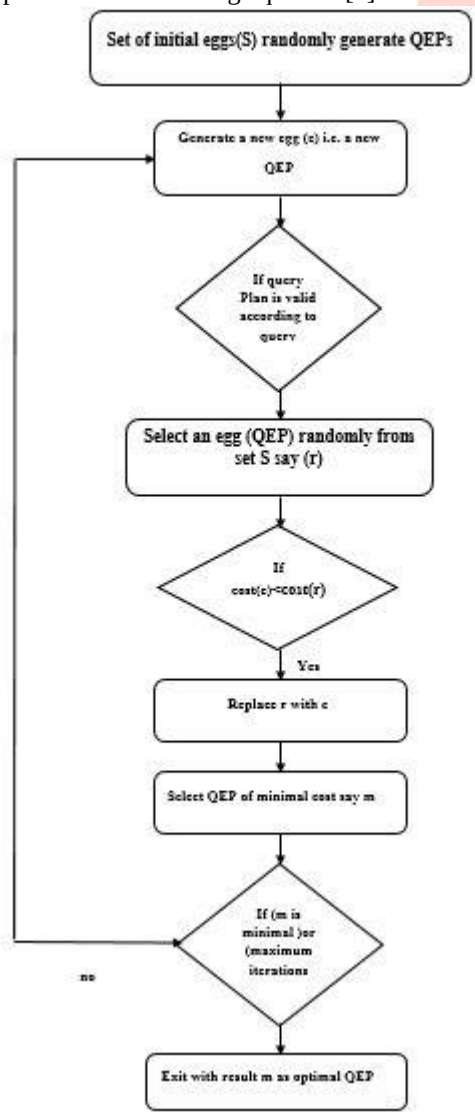
(i.e. Avoid Cartesian Product etc.). After generating a new QEP, one QEP say r is randomly selected from S. We compare the cost of e and r (i.e. similar to comparing eggs), if the cost of e is less than the cost of r; then r is replaced with e (i.e. r is abandoned). After this step minimal cost QEP is selected from set S. If the minimal solution is optimal than this procedure is stopped here otherwise it will repeat the steps of generating the new QEPs until the maximum number of iterations is completed. Flow diagram of this approach is shown in **"Figure 5"** [3].

This algorithm works fine but there would be some situations where it could be stuck in local minima. To identify such situations Tabu search approach is added with proposed solution; it maintains a Tabu list of recently generated, n number of QEPs and a Tabu variable which takes the count of n consecutive generation of new QEPs which are not able to replace the randomly selected QEP from set S. If in some condition it gets in local minima then with the help of Tabu variable and Tabu list, it can be easily identified. After identifying these situations, levy flight is used to come out from these situations i.e. local minima. Levy flight is nothing but it's a process of random generation of new QEP, here the permutation function P2 is used for it. P2 is same as P1 but the only difference is it will generate permutation after more position shuffles meaning degree of changes in position is more. Proposed solution with Tabu search and levy flight is shown in "**Figure 6**" [3].

D. Simulated Annealing

The annealing process in physics consists of obtaining low energy states of a solid element being heated. Simulated Annealing takes advantage of the Metropolis algorithm used to study equilibrium properties in the microscopically analysis of solids. Specifically the Metropolis algorithm generates a sequence of states for a solid object. Given an element in state i with energy E! a new element in state j is produced, if the difference between energies is below zero, the new state is automatically accepted; otherwise its acceptance will depend on certain probability based on the temperature the system is exposed to and a physic constant known as Boltzmann constant k!. Similarly, the simulated annealing algorithm constructs solutions to combinatorial problems linking solution-generation alternatives and an acceptance criterion. The states of the system can be matched to solutions of the combinatorial problem, and in the same way the cost function of the optimization problem can be seen as the energy cost of the annealing system. Therefore the simulated annealing algorithm starts exposing the system to high temperatures and thus accepting solutions that do not improve previous solutions. By terms of a cooling factor, the temperature starts lowering until it reaches zero where solutions that do not improve its parents are not accepted.

The calculation of the acceptance probability of the simulated annealing algorithm is adopted from the Metropolis algorithm and corresponds to the following equation [2].



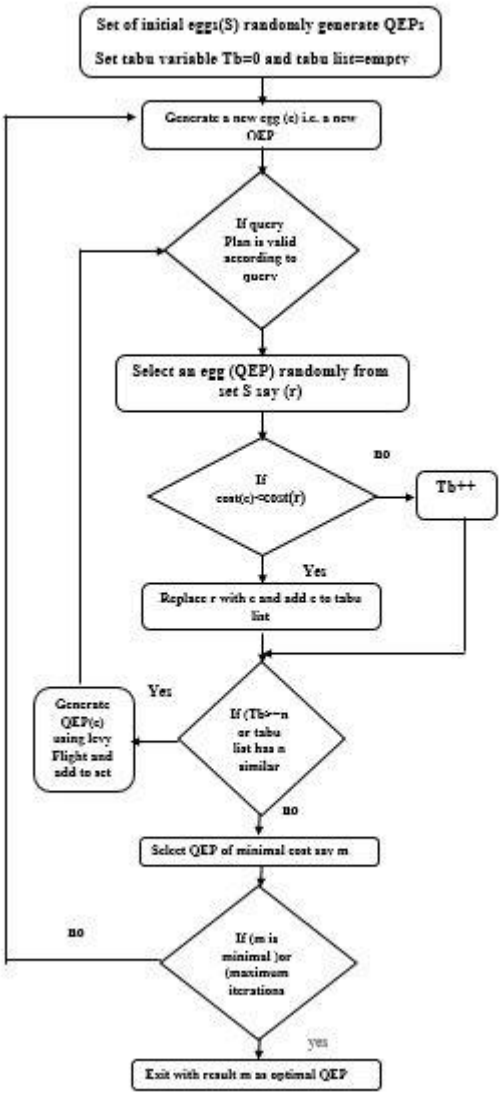


Figure 5: Cuckoo Search algorithm for Join Ordering [3].

Figure 6: Combination of Cuckoo and Tabu Search Algorithm [3]

$$P_T = \begin{cases} 1 & ,if\ f(i) \le f(j) \\ e^{\frac{f(i)-f(j)}{j}} & ,if\ f(j) > f(i) \end{cases} \quad (6)$$

Different implementations of the simulated annealing algorithm have been used to solve the join ordering problem using different cooling schemas, initial solutions, and solution generation mechanisms. The initial state $S!$ was chosen using seminaïve evaluation methods and the initial temperature $T!$ was chosen as twice the cost of the initial state. The termination criterion of the algorithm is composed of two parts: the temperature must be below 1 and the final state must remain the same for four consecutive stages. The generation mechanism is based on a transition probability matrix R: $S! \times S! \rightarrow [0,1]$ where each neighbor of the current state has the same probability to be chosen as the next state

$$R(S,S') = \begin{cases} \frac{1}{|N_A(S)|} & ,if\ S' \in N_A(S) \\ 0 & ,Otherwise \end{cases} \quad (7)$$

Finally authors suggest the use of two different cooling schedules in their implementation. They propose the use of the following equation to control the temperature of the system Tnew =α(Told)Told .The function returns values between 0 and 1. The first strategy proposed consists of keeping a constant value of 0.95 and the second one consists of modifying the value of according to Table 1.

Table 1. Factor to reduce temperature

T0/T≤	$\boldsymbol{\alpha}$
2	0.80
4	0.85
8	0.90
∞	0.95

E. DSQO: Deterministic Swapping Query Optimization

The method proposed as a novel algorithm to solve the traveling sales man problem takes advantage of the automata theory to construct a path to find a global optimal solution to the problem. Specifically, a special type of deterministic finite automaton is constructed to model the solution space of the combinatorial problem and a transition function is designed to allow the navigation around neighbor answers. The exchange deterministic algorithm (EDA) was used to browse the structure to rapidly converge to an optimal solution [2].

a. Query Optimization Based On The Automata Theory

A deterministic finite automaton of swapping, DFAS, is a kind of DFA that allows the modeling of the set of feasible solutions of combinatorial problems where the order of the elements is relevant and no repetitions are permitted. A DFAS is formally defined in (NIÑO and ARDILA, 2009) as a 7-tuple.

$$M = (Q, \sum\ , \delta, q_0, F, X_0, f) \quad (8)$$

Where Q represents the set of all feasible solutions to the problem, Σ is the input alphabet and represents the set of all possible exchanges between two elements of the answer. The author proved that the number of elements of Σ is given by the following equation

$$|\sum\ | = \frac{n \times (n-1)}{2} \quad (9)$$

$\delta: Q \times \sum \rightarrow Q$ is the transition function and takes the node qi $\in Q$ and swap the elements in the positions indicated by the element of the alphabet, $q0$ is the initial state and is given by an initial solution to the problem, F is the set of final states, $X0$ is the input vector containing the initial order of elements corresponding to the state $q0$, f is the objective function of the combinatorial problem that evaluates the given order Xk in the node qk [2].

For instance the following example is given to understand the construction of a DFAS. Given the objective function of a combinatorial optimization problem and an input vector.

$$f(\bar{X}) = 0.3x_1 + 0.2x_2 + 0.1x_3 \quad (10)$$

$$\overline{X_0} = (!,2,3) \quad (11)$$

The alphabet contains six elements consecutively by the application of the definition.

$$\sum = \{(1,2),(1,3),(2,3)\} \quad (12)$$

The transition function is constructed by labeling the state $q0$ after the input vector $\overline{X_0}$ the swap operation is applied to $q0$ for every element of the alphabet and every new vector $\bar{X}_i$ constitutes a new state qi that is included in the DFAS; the process is repeated until every node qi has been evaluated. Table 2 shows the transition function for the given example.

Table 2. DFAS transition Function example

$\delta(q_0,(1,2))$ $= (2,1,3)$ $X_1 = q_1$	$\delta(q_0,(1,3))$ $= (3,2,1)$ $X_2 = q_2$	$\delta(q_0,(2,3))$ $= (1,3,2)$ $X_3 = q_3$
$\delta(q_1,(1,2))$ $= (1,2,3)$ $X_0 = q_0$	$\delta(q_1,(1,3))$ $= (3,1,2)$ $X_4 = q_4$	$\delta(q_1,(2,3))$ $= (2,3,1)$ $X_5 = q_5$
$\delta(q_2,(1,2))$ $= (2,3,1)$ $X_5 = q_5$	$\delta(q_2,(1,3))$ $= (1,2,3)$ $X_0 = q_0$	$\delta(q_2,(2,3))$ $= (3,1,2)$ $X_4 = q_4$
$\delta(q_3,(1,2))$ $= (3,1,2)$ $X_4 = q_4$	$\delta(q_3,(1,3))$ $= (2,3,1)$ $X_5 = q_5$	$\delta(q_3,(2,3))$ $= (1,2,3)$ $X_0 = q_0$
$\delta(q_4,(1,2))$ $= (1,2,3)$ $X_3 = q_3$	$\delta(q_4,(1,3))$ $= (2,1,3)$ $X_1 = q_1$	$\delta(q_4,(2,3))$ $= (3,1,2)$ $X_2 = q_2$
$\delta(q_5,(1,2))$ $= (1,2,3)$ $X_3 = q_3$	$\delta(q_5,(1,3))$ $= (2,1,3)$ $X_1 = q_1$	$\delta(q_5,(2,3))$ $= (3,1,2)$ $X_2 = q_2$

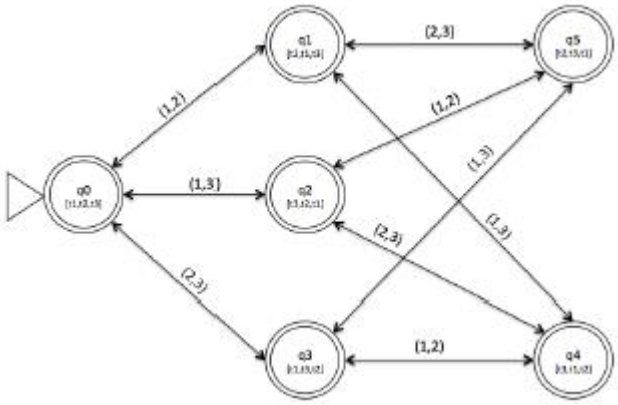


Figure 7: DFAS transition diagram for the example query [3].

b. The Exchange Deterministic Algorithm (EDA)

EDA is a simple algorithm that describes a strategy to browse a DFAS structure to find global optimal solutions to combinatorial problems that allows finding the state that contains the global optimal solution of the problem in polynomial time by only exploring the necessary states minimizing the use of computer memory.

Taking into consideration the characteristics mentioned above the following algorithm was proposed. Where σ is the current state, $\overline{X_\sigma}$ is the vector associated to the current state and f ($\overline{X_\sigma}$) is the value of evaluating the current state´s vector in the objective function.

Step 1: $\sigma = q_0$

Step 2:$\varphi =$ f ($\overline{X_0}$) and $\theta = empty$

Step 3:$\forall a_1 \in \sum$, evaluate $\gamma_1 = f\left(\delta(\sigma, a_1)\right)$ and if $\gamma_1 < \varphi$, let $\varphi = f\left(\delta(\sigma, a_1)\right)$ and $\theta = a_1$

Step 4: If$\theta = empty$, σ is a global optimum. Otherwise $\sigma = \delta\ \sigma$, θ and loop back to step 2.

It is easy to observe that the proposed search strategy does not require the construction of the complete DFAS structure at once. Instead, it only constructs the required areas of the solution space as the neighbors are chosen following the objective function improvement. This strategy is expected to save computer memory because it compares states one by one and rapidly discards portions of the solution space that do not improve the objective function [2].

c. DSQO: Deterministic Swapping Query Optimization

A DFAS structure can be constructed to represent the solution space of the join ordering problem because in fact, it is a combinatorial problem where the order of the elements is relevant and no repetitions are allowed. Following the definition, a DFAS to solve the query optimization problem is the following 7-tuple.

$$M = (Q, \sum\ , \delta, q_0, F, X_0, f(X_i)) \qquad (13)$$

Where Q represents the set of all possible join orders, Σ represents all possible exchanges between two tables in the left deep join query three, δ is the transition function from one query plan to another with a symbol of Σ, $q0$, is a random element of Q selected as the initial state of the automaton, F, is the same set as Q because every plan in Q represents a solution, $X0$, is the vector that contains the order in $q0$ and $f(X\text{i})$, is the objective function that estimates the cost of executing a given Xi plan. The solution space that a DFAS can represent is reduced to all possible left deep trees and thus no bushy tree strategy can be directly explored by this method. To illustrate how to construct a DFAS modeling the join-ordering problem, "Figure 7" shows the transition diagram of the DFAS corresponding to the following query with three tables to join.

```
SELECT *
FROM tab1, tab2, tab3
WHERE tab1.fkt2 = tab2.pk AND
tab2.fkt3 = tab3.pk
```

The transition diagram shows how the solution space of the join ordering problem is represented. Each node of the graph contains a vector with a join ordering in the left deep strategy of the following form:

$$[t1, t2, t3] \rightarrow ((t1 \infty t2) \infty t3)$$

Where the first two elements from left to right are joined first, and then the intermediate table is joined with the next element in the vector, creating another intermediate table. The process is repeated until there are no more elements to join and the last intermediate table contains the expected result.

EDA was proposed as a method to navigate the DFAS structure without building the complete solution space, by the exploration of the neighborhood of a given state and an objective function improvement rule. Even though EDA is capable of finding global optimal solutions to combinatorial problems effectively, it was mainly designed to find optimal solutions to the traveling salesman problem. Despite the similarities between the TSP and the join-ordering problem, it is necessary to adjust the algorithm to perform as well in the solution of the query optimization problem, which is the targeted problem of this work.

The main reason EDA is not efficient in the solution of the join-ordering problem is that the objective function used in the optimization procedure is yet an estimate of the real cost of using a specific join order. Therefore, minimal improvements towards a better solution may not be worth the effort of a new iteration of the algorithm. Another important reason EDA is not effective when applied to the query optimization problem is the lack of representation of the wider solution space, which includes bushy query trees. A slower convergence of the algorithm is caused because it encounters a reasonable number of solutions with Cartesian products in the optimization process of the left-deep only solution space.

In order to improve the convergence speed of the query optimization based on automata theory module, the DSQO algorithm was designed. There were two main improvements made to the original EDA algorithm: an objective function improvement criterion was added in order to avoid unnecessary optimization efforts and a heuristic was included to transform Cartesian product left-deep plans into feasible bushy tree query plans. The heuristic used was taken from the genetic implementation in PostgreSQL.

The objective function improvement criterion added to the algorithm is used to stop the optimization process when the significance of the new optimum found is not relevant in the solution of the problem. An input parameter ∈ [0,1], which can be seen as a threshold value, is required by DSQO to evaluate if the new solution found in the iteration improves the current solution by a certain percentage given by using the following equation.

$$\frac{\varphi - \gamma_i}{\varphi} < \tau \qquad (14)$$

Where is the new solution and is the current best solution found. The parameter can be considered as an attempt to provide the DBA with a tool to adjust the accuracy of the optimization process based on how deep to look into the search space.

The second strategy included in DSQO is the use of a heuristic to avoid Cartesian products in the left-deep solution space and thus expanding it. The heuristic is taken from the PostgreSQL genetic algorithm implementation and works as follows. The given order is evaluated from left to right. At first every relation is seen as a new branch in the query tree and at the beginning, the first relation constitutes the only branch of the tree. Then, every new branch is tried to merge with an existing one or otherwise it is added as a single relation branch to the query tree. Finally, any existing branch in the tree is forced merge to construct a complete query tree.

Comparing DSQO with other meta-heuristic methods applied in query optimization such as simulated annealing, genetic algorithms and ant colony optimization, the proposed method presents a notorious advantage: it only requires a single input parameter. As mentioned before, even though well-known meta-heuristics have been successfully applied to the query optimization problem, configuring the right input parameters is a delicate task. With only one parameter to configure the DSQO takes an enormous advantage against its competitors.

REFERENCES

[1] Ahmed Khalaf Zager Al saedi, Prov Madya Dr.Rozaida bt.Ghazali and Prof. Dr. Mustafa Bin Mat Deris, "An Efficient Multi Join Query Optimization for DBMS Using Swarm Intelligent approach," Information and Communication Technologies (WICT),DOI: 10.1109/WICT.2014.7077312, pp.113-117,IEEE, Dec 2014.

[2] Mukul Joshi and Praveen Ranjan Srivastava, "Query Optimization: An Intelligent Hybrid Approach using Cuckoo and Tabu Search," International Journal of Intelligent Information Technologies, DOI: 10.1155/2014/727658,IJIIT,2013.

[3] Miguel Rodríguez, Daladier Jabba, Elias Niño, Carlos Ardila and Yi-Cheng Tu, "Automata Theory Based Approach to the Join Ordering Problem in Relational Database Systems," 2nd International Conference on Data Management Technologies and Applications, DOI: 10.5220/0004433802570265, Research Gate, Sep. 2013.

[4] HamidReza Kadkhodaei and Fariborz Mahmoudi, "A combination method for Join Ordering Problem in relational databases using Genetic Algorithm and Ant Colony," Granular Computing (GrC), 2011 IEEE International Conference, DOI: 10.1109/GRC.2011.6122614, pp. 312-317, IEEE, Nov 2011.

[5] Swati V. Chande and Madhavi Sinha, "Genetic Optimization for the Join Ordering Problem of Database Queries," India Conference (INDICON), 2011 Annual IEEE, DOI: 10.1109/INDCON.2011.6139336, pp. 1-5, IEEE, Dec. 2011.

[6] S. Vellev, "Review of Algorithms for Join Ordering Problem in Databse Query Optimization," Information Technologies and Control,Research Gate,2011.