Research Journal of Information Technology

ISSN 1815-7432

Research Journal of Information Technology 2 (3): 139-144, 2010
ISSN 1815-7432

Query Optimization Using Genetic Algorithms

[1]M. Sinha and [2]S.V. Chande
[1]Birla Institute of Technology, Jaipur Campus, Jaipur, India
[2]International School of Informatics and Management, Jaipur, India

Abstract: Query optimization is essentially a complex search task - a search for the best possible plan from among the semantically equivalent plans that can be generated for any given query. It therefore seems logical to consider query optimization in terms of search algorithms. Various search algorithms have been applied by researchers to find an optimal plan for query execution. With queries getting more and more complex the search complexity is increasing. Exhaustive techniques are adequate for trivial instances only, while combinatorial optimization techniques are vulnerable to the peculiarities of specific instances. Current query optimization techniques are thus, inadequate to support some of the emerging database applications. Genetic Algorithms (GAs) are becoming a widely used and accepted method for very difficult optimization problems. This study reviews the studies carried out on application of Genetic Algorithms to Database Query Optimization. The application of Genetic Algorithm to query optimization is motivated by GA's robustness and efficiency in a wide area of search problems. From the studies reviewed it turns out that Genetic Algorithms are a viable alternative to existing query optimizers for optimization of very large queries.

Key words: Database, query optimization, genetic algorithms, join order optimization, genetic query optimization

INTRODUCTION

Genetic Algorithm is a probabilistic algorithm simulating the system of natural selection of living organisms and is often used to solve problems having expensive solutions. In GA, the search space is composed of candidate solutions to the problem. Each solution, usually represented by a string, is termed as a chromosome. Each chromosome has an objective function value, called fitness. A set of chromosomes together with their associated fitness is called the population. This population, at a given iteration of the genetic algorithm, is called a generation. Holland, De Jong and Goldberg were pioneers of GA in the context of continuous non-linear optimization (Holland, 1975; DeJong, 1975; Goldberg, 2003). Chande (2004) has reviewed the applications soft computing techniques including genetic algorithms, to decision making and hence searching from a search space of different alternatives. Applications of GAs in various domains have been discussed by Chande and Sinha (2008a).

Application of GAs is not new to the database query optimization problem. Quite a few studies have been carried out in this field and these studies indicate the viability of GA application to the problem. Chande and Sinha (2008b), done the comparative

Corresponding Author: Dr. Madhavi Sinha, Department of Computer Science, Birla Institute of Technology, 27 Malviya Industrial Area, Jaipur-302017, India Tel: 91-141-4019814

study of the use of Genetic Algorithms for optimization of relational database queries. This study attempts to gain an exhaustive view of improvement in database query optimization using genetic techniques.

Query Optimization Process

Query optimization is the task of improving the strategy for processing a database query. It thus forms an important step in query processing. Query processing refers to the range of activities involved in extracting data from a database. These activities include translation of queries into expressions that can be implemented at the file system's level since these queries are submitted to the DBMS in a high level language, query optimization steps, transformations and query evaluation.

A database query on relational databases can be represented as a query tree, where the leaf nodes represent accesses to relations. The intermediate nodes process and combine the data from their input nodes using physical implementations of the relational operations of project, join etc. and the root node returns the results. Figure 1 illustrates the path from a user query to execution plan (query tree representation).

As the number of relations from which data is sought goes up, the number of alternative solutions for the query also increases drastically. Query optimization can hence be reduced to a search problem where the DBMS needs to find the optimum execution plan in a vast search space. Each execution plan can be considered a possible solution for the problem of finding a good access path to retrieve the required data, every execution plan is an individual (member) in a population.

Genetic Algorithms

Inspired by natural selection, genetic algorithms are abstractions of biological evolution and are thus a method for moving from one population of chromosomes to a new population

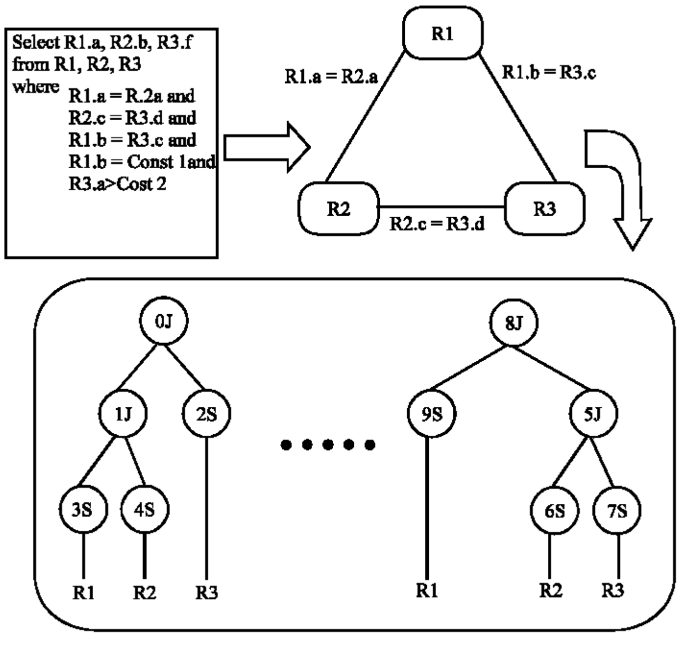

Fig. 1: Path from user query to execution plan

by using a kind of natural selection together with genetics inspired operators of recombination, mutation and inversion (Mitchell, 1998).

In Genetic Algorithm, the solutions are called individuals or chromosomes. After the initial population is generated randomly, selection and variation function are executed in a loop until some termination criterion is reached. Each run of the loop is called a generation. The selection operator is intended to improve the average quality of the population by giving individuals of higher quality a higher probability to be copied into the next generation. The quality of an individual is measured by a fitness function. A fitness function prescribes the optimality of a solution (that is, a chromosome) in a genetic algorithm so that particular chromosome may be ranked against all the other chromosomes.

Crossover and mutation are important operators of the genetic algorithm. Crossover selects genes from parent chromosomes and creates a new offspring. After a crossover is performed, mutation takes place. This is to prevent falling of all solutions in a population into a local optimum of solved problem. Mutation randomly changes the new offspring.

Genetic Algorithms for Query Optimization

When Genetic Algorithms are applied to query optimization, the initial population is generated randomly. For each generation, genetic operations are carried out and thus the population evolves, usually decreasing the average cost of its individuals. When the most optimal plan is obtained, it is passed to the database engine to execute the query.

Optimization of queries can be done through two approaches, one consisting of algebraic manipulations or transformations and the other including strategies to take advantage of the storage of the relations. The algebra based optimization approach is to first represent each relational query as a relational algebra expression and then transform it to an equivalent but more efficient relational algebra expression. The transformation is guided by heuristic optimization rules. The basic idea of the cost-estimation-based approach is - For each query, enumerate all possible execution plans. For each execution plan, estimate the cost of execution plan (Hector *et al.*, 2004). Finally choose the execution plan with the lowest estimated cost (Raghu and Johannes, 2000). Enumerative strategies can lead to the best possible solution, but face a combinatorial explosion for complex queries (e.g., a join query with more than ten relations. Join operation is not only frequently used but also expensive (Steinbrunn *et al.*, 1997)). In order to investigate larger spaces, randomized search strategies have been proposed (Horng *et al.*, 2000) to improve a start solution until obtaining a local optimum. Examples of such strategies are simulated-annealing (Ioannidis and Wong, 1987) and iterative-improvement (Swami and Gupta, 1988). With the same objective, genetic search strategies (Goldberg, 2003) can be applied to query optimization, as a generalization of randomized ones. Randomized or genetic strategies do not guarantee that the best solution is obtained, but avoid the high cost of optimization. As an optimizer might face different query types (simple vs. complex) with different requirements (ad-hoc vs. repetitive), it should be easy to adapt the search strategy to the problem (Lanzelottel and Patrick, 1991).

The major issue in query optimization is that, the search space is complicated and genetic algorithms are theoretically and empirically proven to provide robust search in complex spaces. These algorithms are computationally simple yet powerful in their search for improvement. They are not fundamentally limited by restrictive assumptions about search space (Goldberg, 2003). The use of GA approach in addressing the Query Optimization issue, therefore seems appropriate.

Genetic algorithms may be employed in obtaining an optimal solution for each of the two approaches. They may contribute towards the selection of an efficient relational algebra expression and may also find near-optimal execution plans through efficient cost estimation.

A Survey of Database Query Optimization and Genetic Algorithms

All query optimization algorithms primarily deal with joins. Most studies on the use of Genetic Algorithms in Query Optimization also thus focus on joins. Selection of appropriate index for query execution is also one of the major concerns and hence substantial research has also been done in the use of Genetic Algorithms for index selection.

Kristin *et al*. (1991) have studied genetic algorithms for join query optimization. They have given a method for encoding arbitrary binary trees as chromosomes and describe several recombination operators for such chromosomes. Their performance results show that genetic algorithms can effectively identify high quality query execution plans and the selected plans are in general comparable to or better than the current best-known method for query optimization particularly, the output quality and the time needed to produce such solutions.

Fotouhi and Galarce (1991) of Wayne State University Computer Science Department have proposed using genetic algorithms to search for near-optimal indexing. Fotouhi and Galarce (1991) experiment gave encouraging results but was based on a single table, a rare occurrence in the real world.

Celko (1993) extended the Fotouhi-Galarce experiment to work on multiple tables. He combined columns to make tables using normal forms built from Functional Dependencies. He used a sample database, identified the functional dependencies for the database and created a query chromosome structure having genes based on attributes.

Since, the operations on the schema were complex than those used for indexes on a single table by Fotouhi and Galarce (1991) tables had to combine or split. The goal was to have the smallest number of tables used in the queries to avoid the cost of joins. Once the tables were determined for the set of queries, the index genetic algorithm was applied to the tables (Celko, 1993).

Kratica *et al*. (2003) have proposed a genetic algorithm (GA) for solving the ISP (Index Selection Problem) i.e., the problem of minimizing the response time for a given database workload by a proper choice of indexes. Their GA is based on binary encoding, data structures for the evaluation of the objective function, on the uniform crossover and simple mutation. They have tested the algorithm on the class of challenging instances known from the literature and demonstrate that the results obtained indicate its efficiency and reliability.

Utesch's (1997) model attempts to find the solution of the QO problem similar to a Traveling Salesman Problem (TSP). He has used Postgres Query Optimizer for the research. The GEQO module allows the Postgres query optimizer to support large join queries effectively through non-exhaustive search.

Lanzelottel and Valduriez (1991) have given a solution to the extensibility of the query optimizer search strategy. This solution is based on the object-oriented modeling of the query optimizer, where the search space and the search strategy are independently specified. It is illustrated by applying different search strategies including the genetic algorithm approach.

Steinbrunn *et al*. (1997) have studied different algorithms that compute approximate solutions for optimizing join orders. They extensively scrutinized optimizers from the three classes, heuristic, randomized and genetic algorithms. From their study it turns out that randomized and genetic algorithms are well suited for optimizing join expressions.

The studied several algorithms for the optimization of join expressions and inferred that randomized and genetic algorithms are much better suited for join optimizations; although they require a longer running time, the results are far better.

More recently, Fang *et al*. (2008) proposed a novel multi-copy join optimization method based on genetic algorithm to accomplish global query rewriting by taking redundancies into account. The proposed join optimization method considers both choosing redundant copies of a global view and searching a relative low-cost join order by its encoding schema and special genetic operators.

Research studies have also been carried out on the implementation of Genetic Programming rather that Genetic Algorithms to database query optimizations by Stillger and Spiliopoulou (1996) and Muntes-Mulero *et al.* (2006) .

Chande and Sinha (2008c), the authors have analyzed the previous studies on the application of genetic algorithms to query optimization and presented a framework for a genetic query optimizer. They have also carried out a comparative analysis of the genetic join order optimizers on various parameters. The parameters used in the comparison include the GA type used, encoding techniques, solution space, initial population generation, selection techniques, fitness functions applied, crossover and mutation types, number of generations, sizes of queries, crossover, mutation, population and termination criterion.

CONCLUSIONS

To a large degree, the success of a database management system lies in the quality, functionality and complexity of its query optimizer, since that determines much of the system's performance. GAs are characterized by higher probability of finding good solutions for large and complex problems which query optimization certainly is.

Overall, the results of the studies unconditionally prove the applicability of genetic optimization algorithms to the problem. GAs may prove to be a competitive alternative to deterministic optimization algorithms even for small solution spaces.

The results of the attempts to apply Genetic Algorithms to query optimization so far have thus been very encouraging.

REFERENCES

Celko, J., 1993. Genetic algorithms and database indexing. Dr. Dobb's J., April 1993.

Chande, S.V. and M. Sinha, 2008a. Genetic algorithms: A versatile optimization tool. Int. J. Inform. Technol., 1: 7-13.

Chande, S.V. and M. Sinha, 2008b. Optimization of relational database queries using genetic algorithms. Proceedings of the International Conference on Data Management, 2008, IMT Ghaziabad.

Chande, S.V. and M. Sinha, 2008c. The use of genetic algorithms for optimization of relational database queries: A comparative study. Proceedings of the IEEE Sponsored National Conference on Applications of Intelligent Systems-2008, Hindu College of Engineering, Sonepat.

Chande, S.V., 2004. Decision making using soft computing techniques. Proceedings of the Souvenir of the National Seminar on Emerging trends in Soft Computing based Artificial Intelligence, MBM Engineering College, JNV University, Jodhpur, Feb. 27-29, 2004.

DeJong, K.A., 1975. An analysis of the behavior of a class of genetic adaptive systems. Ph.D. Thesis, University of Michigan.

Fang, L., P. Wang, J. Yan, 2008. A multi-copy join optimization of information integration systems_based on a genetic algorithm. Proceedings of the 2008 3rd International Multi-Conference on Computing in the Global Information Technology, July 27-Aug. 01, Washington, DC, USA., pp: 223-228.

Fotouhi, F. and C.E. Galarce, 1991. Genetic algorithms and the search for optimal database index selection. Lecture Notes Comput. Sci., 507: 249-255.

Goldberg, D.E., 2003. Genetic Algorithms in Search, Optimization and Machine Learning. Pearson Education, New Delhi, ISBN-10: 0201157675.

Hector, G.M., D.U. Jeffrey and W. Jennifer, 2004. Database Systems: A Complete Book. Pearson Education Singapore Pvt. Ltd., Singapore.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. 1st Edn., University of Michigan Press, Ann Arbor, Michigan, ISBN: 0472084607.

Horng, J.T., C.Y. Kao and B.J. Liu, 2000. A genetic algorithm for database query optimization. Int. Conf. Evol. Comput., 1: 350-355.

Ioannidis, Y.E. and E. Wong, 1987. Query optimization by simulated annealing. ACM SIGMOD Rec., 16: 9-22.

Kratica, J., I. Ljubiæ and D. Tosic, 2003. A genetic algorithm for index selection problem. Lecture Notes Comput. Sci., 2611: 281-291.

Kristin, B., M.C. Ferris and Y. Ioannidis, 1991. A Genetic Algorithm for Database Query Optimization. University of Wisconsin, Madison.

Lanzelottel, R.S.G. and P. Valdureiz, 1991. Extending the search strategy in a query optimizer. Proceedings of the 17th International Conference on Very Large Databases, Sept. 03-06, Morgan Kaufmann Publishers Inc. San Francisco, pp: 363-373.

Mitchell, M., 1998. An Introduction to Genetic Algorithms. The MIT Press, USA., ISBN-10: 0262631857, pp: 221.

Muntes-Mulero, V., J. Aguilar-Saborit, C. Zuzarte and J.L. Larriba-Pey, 2006. CGO: A sound c graphs. Proceedings of the International Conference on Computational Science, May 28-31, Springer-Verlag, The University of Reading, UK., pp: 156-163.

Raghu, R. and G. Johannes, 2000. Database Management System. 2nd Edn., McGraw Hill, Singapore.

Steinbrunn, M., G. Moerkotte and A. Kemper, 1997. Heuristic and randomized optimization for the join ordering problem. VLDB J., 6: 191-208.

Stillger, M. and M. Spiliopoulou, 1996. Genetic programming in database query optimization. Proceedings of the 1st Annual Conference on Genetic Programming, July 28-31, Stanford, California, pp: 388-393.

Swami, N. and A. Gupta, 1988. Optimization of large join queries. ACM SIGMOD Rec., 17: 8-17.

Utesch, M., 1997. Genetic Query Optimization in Database Systems. Institute of Automatic Control, University of Mining and Technology in Freiberg, Germany.