Randomized Optimization of Quantified Path Expressions in Object Databases

Zhao-Hui Tang — Georges Gardarin

CNRS-PRiSM Laboratory
University of Versailles-St-Quentin
78035 Versailles, France
{tang, gardarin}@prism.uvsq.fr

ABSTRACT. Object relational and object database (distributed) systems are used more and more in real applications due to their rich and extensible data types and query language mixing declarative assertions and navigations. Queries may frequently involve multiple collection traversals (i.e., path expression), possibly distributed. Moreover, predicates with quantifiers can also be included inside path expressions. These bring new challenges for object query optimization. In order to efficiently optimize these queries, we define the concept of quantified path expressions and introduce navigational algebraic operators, which mix nicely with classical object algebra. We also investigate several randomized algorithms for exploring the search space generated by path expressions. Through experimentation, we also find an improved procedure of Genetic Algorithm which is more efficient compared to the classical one. We give both analytical and experimental comparisons between our Genetic Algorithm and other well-known combinatorial algorithms such as Iterative Improvement, Simulated Annealing and Tabu Search. For most of the experiments, our Genetic Algorithm finds better query plans than other algorithms.

KEY WORDS: object databases, query processing, path expression, optimization algorithm, genetic algorithm.

1. Introduction and Motivations

In federated object-oriented database systems, traversing efficiently a sequence of relationships appears to be an important feature. Due to the sophisticated integration of various data sources, complex queries are generated which often traverse multiple collections distributed within the network. This has been verified both for telecommunication applications using the CMIS protocol [FFL 97] and for Web-based query languages generalizing navigation within query graphs [Sue 96]. However, even in centralized object DBMSs, path traversals are often poorly optimized. Only set-oriented or Pipelined binary forward joins of collections are considered. Various sorts of navigational procedures are generally not optimized in current products (as it was in network DBMSs). Considering navigational algorithms gives the possibility of stopping early and becomes very important in distributed object systems with distributed relationships, as IRO-DB [GGF 96]. Finding the best query plan with both set-oriented and navigational algorithms is not an easy task.

To make it possible, the exploration of query plans with object navigations, an extension of object algebra is necessary, especially with navigational operators. Object algebra has been focused on several years and many variations can be found in research literature [AFS 89], [BLD 89], [VD 91]. Most of these proposals are extensions of the relational algebra, by redefining certain aspects of the operators and suggesting several extensions to deal with nested attributes, methods, inheritance, etc. To introduce some sort of navigation, [SC 89] proposes different pointer-based joins and compares their performance. The OID reference attribute is used as the join attribute, and since the OID value represents a physical object location (in the case of physical OID), it may be profitable to sort these OID values before accessing objects to avoid redundant disk page accesses. This opens the way to new join method implementations (pointer-based joins) [DLM 93]. Although the new algorithms profit from certain OID techniques, it is still a set-oriented join. In [BMG 93], a logical algebra named materialize is proposed to assemble complex objects. Although the paper only studies binary pointer chasing algorithms, it points out a research direction in which more logical navigation operators need to be investigated for optimizing object queries in a better way.

![Figure 1. A database schema](image-url)
Let us consider a database schema as presented in Figure 1. A sample query is "find all the papers on 'query processing' whose authors are all less than 30 years old". There are three collections involved in this query: paper, author, and lab; also there are quantifiers involved in the path. How to efficiently process such a query is still an open problem. It can be evaluated by using a sequence of traditional set-oriented join operators as well as by navigation. Navigation can also be performed using different strategies, such as depth-first, breadth-first, best-first graph traversals. In [GGT 96], a performance study of pointer-based navigation versus set-oriented joins for evaluating path expressions is presented; the result shows that object navigation and set-oriented join should coexist in an efficient ODBMS as no one dominates the others.

The problem of processing path expressions is similar to processing a sequence of joins in relational system; however, direct OIDs and quantifiers introduce new algorithms based on direct navigation and improved stop conditions. It becomes crucial to explore the extended query space in an efficient way to improve query performance. When the number of algorithms increases, the search space increases exponentially; its precise size has already been studied in [OL 90], [IK 91], [TL 91], [LVZ 93]. Due to the large size of the search space, it is obvious that the query optimizer cannot explore all the candidate query execution plans (QEP) with an exhaustive method. Combinatorial algorithms such as Simulated Annealing, Iterative Improvement [NSS 86], [IW 87], [SG 88], [IK 90] and Tabu Search [GLO 89], [GLO 90] have been introduced to solve this problem. These strategies are based on a set of transformation rules and also a cost model for measuring the effectiveness of execution plans. Their performance depends on the cost distribution in the search space and on the set of transformation rules being used. Thus incompleteness of transformation rules and non continuous search space will result in missing good plans.

Simulated Annealing and Tabu Search strategies start from an initial plan, and keep on evaluating by applying transformation actions. Thus the initial plan plays an important role: different initial plans can lead to different local minimums. Unfortunately, the costs among different local minimums can be radically different [LVZ 93]. To overcome the unique initial plan limitation, [IK 90] cleverly proposes to combine Iterative Improvement and Simulated Annealing in a new strategy called two-phase optimization coupling Iterative Improvement and Simulated Annealing. A similar approach is proposed in [LVZ 93] with Toured Simulated Annealing, where several Simulated Annealing tours with different starting points are performed. However, the implementation of combined search algorithms in a real system demonstrates the complexity of such approaches [FV 95] for little profits. [GPK 94] proposes a transformation free strategy. Their idea is to randomly generate execution plans instead of using transformation rules. Their approach provides a cheap and effective alternative to transformation-based algorithms. But
the technique relies on an efficient mechanism to generate query plans uniformly distributed over the search space, which is not a trivial task.

The adaptation of a Genetic Algorithm to the query optimization problem was first proposed in [BFI 91] for exploring the search space of join query plans. This class of methods is based on the principles of natural selection and natural genetics that combine the notion of survival of the fittest, random and yet structured search, and parallel evaluation of candidates in the search space. It never works on one particular plan, but on a population of plans. [BFI 91] investigates both the left-deep tree space and the bushy tree space using two binary join algorithms (nested-loops and merge-scan). The results demonstrated that the proposed GA algorithm performs quite well in comparison to Dynamic Programming (the classical System R method) for large queries, with more than 15 joins. The Genetic Algorithm was recently revisited in [SMK 97]. The authors compare most of the deterministic and randomized algorithms. They demonstrate that: (i) deterministic algorithms are only feasible for queries that are not too complex (less than 6 to 7 relations for bushy trees); (ii) randomized and genetic algorithms operating in the bushy tree solution space are the most appropriate strategies when the problems are too complex to be tackled by exhaustive enumeration. Based on their excellent results, it appears that randomized algorithms are the best suited for exploring the search space of complex path expressions traversing more than 6 to 7 collections.

In this paper, we first introduce quantified path expression, i.e., path expressions involving selection predicates with various types of quantifiers. We then propose an extended object algebra to express various kinds of navigational operators and combine them with classical set-oriented algebraic operators. Thus, considering both set-oriented and navigational bushy trees with several nary join algorithms makes the search space of query plans to investigate very large. To explore it and determine as much as possible the best plan, we compare four randomized algorithms, namely our own variation of Genetic Algorithm (GA), Simulated Annealing (SA), Iterative Improvement (II), and Tabu Search (TS). The results demonstrate the superiority of the proposed GA for path expressions involving at least 10 collections. The new contributions of this paper are: (i) a generalization of path expressions to complex predicates with quantifiers; (ii) a definition of algebraic operators that captures navigational algorithms for processing complex paths; (iii) an experimental evaluation of the genetic algorithms against other randomized algorithms, which shows the superiority of the former.

The paper is set out as follows. Section 2 introduces quantified path expressions and new path traversal operators. Section 3 presents the framework of our application of Genetic Algorithm for optimizing path expressions. Section 4 shows the experimentation results of different randomized algorithms, including Tabu Search that has not been so far experimented within literature. Section 5 concludes the paper.