Modeling genetic methods for data search strategies with ANOVA

Victor Muntés-Mulero\textsuperscript{1}, Marta Pérez-Cassany\textsuperscript{2}, Josep Aguilar-Saborit\textsuperscript{1} and Josep-Lluís Larriba-Pey\textsuperscript{1}

\textsuperscript{1} Departament d’Arquitectura de Computadors (DAC). Universitat Politècnica de Catalunya (UPC). C/Jordi Girona 1-3 08034 Barcelona, Spain. \{vmuntes, jaguilar, larri\}@ac.upc.edu

\textsuperscript{2} Departament de Matemàtica Aplicada II. UPC. marta.perez@upc.edu

Abstract: Database management systems (DBMS) must find the most efficient strategies in order to retrieve the data required by user queries. However, with the increasing need for storing huge amounts of information, traditional optimization based on exhaustive search methods fail to find the optimal strategies. Genetic Programming has been proposed as an alternative random search strategy to improve the optimization quality in these situations. In this paper, we show that the ANOVA technique is useful to analyze DBMS optimization efficiency. We propose a statistical model fitted to 7290 observations that is able to predict the cost of the best result obtained by the optimization process, using an optimizer for DBMSs based on genetic approaches. Also, it offers the possibility to establish criteria to configure a genetic optimizer in order to obtain the best possible results.

Keywords: query optimization; genetic optimization; ANOVA; nested factors; random effects.

1 Introduction

Large amounts of data pose special problems for Knowledge Discovery in Databases. Specifically, deciding the most efficient method to retrieve the data required by a user is still an unresolved problem, when the amount of relations involved in a single query is very large. In the Database Management Systems (DBMS) environment, this problem is usually referred as the large join query optimization problem [Ono, 90]. Special attention is payed to this issue by proposing new methods to find near-optimal search strategies for a fast data retrieval. One of the most recent proposals is based on genetic programming [Banzhaf, 98] search strategies. The suitability of evolutionary strategies has been shown to deeply depend on the problem to be solved. Due to the complexity of DBMSs, deciding whether an algorithm based on genetics is appropriate or not cannot be decided at a glance and demands a more formal statistical analysis.
In this paper, we show that the ANOVA technique [Scheff, 1959] [Montgomery, 1991] is very useful in this scenario. The linear model is based on 7290 observations, each one calculated from the average of 10 executions obtained under the same conditions, and it allows for 8 different factors. With this analysis, we prove that it is possible to find a model that predicts the average best cost obtained by a genetic optimizer. This implies that we can accurately study the effect of the different factors involved in the optimization process and set the base to establish criteria to parameterize a genetic optimizer.

In what follows, we describe the organization of the rest of the paper. Section 2 briefly describes our genetic optimizer. Section 3 presents the statistical model. Finally, Section 4 concludes the paper.

2 The genetic optimizer

Given a user query, typically written in SQL (Structured Query Language), a DBMS optimizer is in charge of deciding the best way to access the relations in the database, in order to retrieve the required information as fast as possible. The output of an optimizer is usually called query execution plan (QEP) (Figure 1). A query execution plan is commonly represented by a directed binary tree structure. The leaf nodes are the scan operations used to access the relations in the database, and the non-leaf nodes usually represent joining operations that merge data coming from two data sources. The root node returns the final results. In order to evaluate the goodness of a QEP, the optimizer uses a very complex cost function that express the fitness of the solution. The cost model used for this experiment can be found in [Muntés, Dec05].

![Query Execution Plan (QEP) example.](image)
As the number of relations involved in the query increases, the number of possible execution plan increases exponentially. Traditional optimizers cannot handle this situation since they are based on exhaustive search through the search space. Genetic optimizers, based on evolutionary strategies, have been proposed to remedy this situation.

The genetic optimizer used in this study is called CGO [Muntés, Apr05] and it is based on genetic programming. CGO uses the same execution patterns of any evolutionary algorithm. Starting with an initial population \( P \) containing a known number of members denoted by \( N \), usually created at random from scratch, two operations are used to produce new members in the population: crossover operations, which combine properties of the existing members in the population and mutation operations, which introduce new properties into the population. In order to keep the size of the population constant, a third operation, usually referred as selection, is used to discard the worst fitted members, using a fitness function. This process generates a new population, also called generation, that includes both the old and the new members that have survived to the selection operation. This is repeated iteratively through a fixed number of generations denoted by \( G \). At the end, the best solution is chosen from the final population. We call \( C \) the number of crossovers executed per generation and \( M \) the number of mutations. Deciding the best value for each of these parameters is still an unresolved problem. The model presented in the next section offers a practical solution for deciding their impact and, therefore, establish criteria to parameterize a genetic optimizer.

3 General Statistical Model

In this section, we study the effect of different categorical variables on the average cost of the best QEP obtained after an execution by means of the Analysis of Variance technique. We have used the Statistical Analysis System (SAS) Release 8.00 [SAS, 99].

3.1 Variables in the model

Our statistical model aims at predicting the expected average cost of the returned QEP depending on different factors. As a consequence, the average cost of the returned QEP is the dependent variable. The factors used in the model are summarized in Table 1. Variables \( N, G, C \) and \( M \) have been fixed to take just three different values. In consequence, they can be considered factors. Our model must be general for any query and, therefore, it must consider some variables that define the main aspects of the query structure. Therefore, besides \( N, G, C \) and \( M \), we must consider two more independent variables of fixed effects which are: the number of Relations involved in the query \( (R) \) and the selectivity factor of the restrictions in the query \( (S) \).
TABLE 1. Independent factors in the model.

<table>
<thead>
<tr>
<th>Factors</th>
<th>Description</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td># of relations involved in the query</td>
<td>fixed</td>
</tr>
<tr>
<td>S</td>
<td>Selectivity of the query</td>
<td>fixed</td>
</tr>
<tr>
<td>Q</td>
<td>Query</td>
<td>random</td>
</tr>
<tr>
<td>N</td>
<td># of members in the population</td>
<td>fixed</td>
</tr>
<tr>
<td>P</td>
<td>Population</td>
<td>random</td>
</tr>
<tr>
<td>G</td>
<td># of generations</td>
<td>fixed</td>
</tr>
<tr>
<td>C</td>
<td># of crossovers</td>
<td>fixed</td>
</tr>
<tr>
<td>M</td>
<td># of mutations</td>
<td>fixed</td>
</tr>
</tbody>
</table>

is related to the probability of a tuple (or an item) in a base relation to qualify and to be returned as a result. Thus, queries with restrictions on the attributes that impose a selectivity factor close to 0 will discard, on average, most of the tuples in the base relation, and queries with selectivity factors close to 1 preserve most of the information, increasing the cost of the associated QEPs. It is reasonable to consider the level values for factor N proportional to R, in consequence, N is nested in R. Analogously, we consider the level values for C and M proportional to N and, for that reason, C and M are both nested in N.

Due to the intrinsic randomness of the optimization process, it has sense to assume that there will be important differences in the cost for different queries with the same selectivity and the same number of relations. Thus, the query (Q) will appear in the model as a random effects factor. This allows us to consider the obvious effects of the query itself. The same applies to the initial population (P), and in consequence the model will contain two random effects factors.

3.2 Description of the experiments

All the decisions taken in this section are arrived at based on practical and empirical considerations from experiments done in previous ad-hoc tests, in

TABLE 2. Independent fixed effects factor levels studied in the experiment.

<table>
<thead>
<tr>
<th>Fixed effects</th>
<th>Independent Variables</th>
<th>Studied Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>20 and 50</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>10^{-2}, 10^{-4} and 10^{-8}</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>4R, 8R, and 12R</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>50, 100, and 200</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>N, \frac{N}{2}, and \frac{N}{4}</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>\frac{N}{2}, \frac{N}{4}, and \frac{N}{8}</td>
<td></td>
</tr>
</tbody>
</table>
order to use reasonable and realistic values. Table 2 summarizes the levels used for the different factors in the experiment. Figure 2 depicts the experiment design. For every level in $R$ we randomly create 9 different queries ($Q$), three for each level of $S$. Given a query, we create 15 populations ($P$), 5 for each level of $N$. Once we have created the populations, we run 270 executions. This set of executions is divided into three subsets, corresponding to the three levels in $G$. For every level of $G$, we subdivide the executions into 9 subsets corresponding to the crossing between $C$ and $M$. Finally, for every possible configuration of the levels of each factor we run 10 executions and calculate the average cost obtained from them. Therefore, since we have run 72900 independent executions, the number of observations for the dependant variable is equal to 7290. The executions have been run on a distributed system with 80 nodes. Each node consist of an Intel Xeon 2.8 Ghz and 2 GB of RAM. The total amount of execution time used to obtain the results in this experiment is equivalent to more than 2700 hours in a single processor.

![FIGURE 2. Design for obtaining the sample observations.](image)

### 3.3 Model definition and Goodness of fit

We transform the average cost taking logarithm and we depart from the model that contains all the main effects and all the first-order interactions between the fixed effects factors. The first analysis of the model reveals the existence of 7 outliers. All the outliers correspond to configurations where the number of relations involved in the query is 50 ($R = 50$), the number of members in the population is the minimum tested ($N = 4R$), the number of generations is 50 ($G = 50$) and the number of crossovers and mutations is low. These results make sense since, under this configuration, we ask the genetic optimizer to obtain a good QEP in a huge search space, limiting its search potential by using the minimum number of members, generations
and operations tested and, therefore, it is possible that sporadically some observations stop far from a reasonably good QEP. Once these outliers are removed, the data set has 7283 observations. The $R^2$ obtained with the same model and the new data set is 0.9693. After removing the interactions between $G$ and $M$, and between $C$ and $M$, both nested in $N$, which are not significative, the final model is (1). For this model the values of $R^2$, the Mean Square Error and the coefficient of variation are equal to 0.9691, 0.254 and 1.34, respectively. On the one side, this means that the model explains the 96.91 % of the total variability in the data. On the other side, the fact that the error standard deviation is quite small implies that the predictions will be accurate.

$$\log(\text{Cost}) = R + S + R \ast S + Q(R \ast S) + N(R) + S \ast N(R) + G + P(Q \ast N \ast R \ast S) + C(N \ast R) + M(N \ast R) + R \ast G + S \ast G + S \ast C(N \ast R) + S \ast M(N \ast R) + G \ast N(R) + G \ast C(N \ast R)$$ (1)

Figure 3 shows us the relation between the observed and the fitted values. We can graphically see that our model is very good in predicting the average expected cost, depending on the different factors studied in this paper. Also, we can observe that the concentration of values near to 17 and 18 is higher.
than in other areas. This happens because the optimizer is trying to find a minimum cost. In our experiments, for queries involving 20 relations, the minimum logarithm of the average cost seems to be around 17. The same effect can be observed for queries involving 50 relations, around 18. Figure 4 shows the studentized residuals versus the values fitted by the model. Out of the 7283 residual values, only 521 present an absolute value higher than 1.96, which represents the 7.15% of the total residual values. For the same reason explained above, in this plot we find two marked straight lines with a negative slope.

The hypothesis of independence of a linear model is guaranteed as a consequence of the fact that for a given population of a given query, we run independent executions. With respect to the normality assumption, studentized residuals have a larger kurtosis than the corresponding normal distribution with the same mean and variance.

4 Results and Conclusions

We defined a statistical model that allows us to study the importance of different factors in the optimization process of a DBMS. From the model, we extract the following conclusions:
• The ANOVA technique with fixed and random effects factors is useful to study and model the problem of optimization in DBMSs. We have defined a model that is able to accurately predict the logarithm of the average cost obtained after the optimization process, using a genetic optimizer.

• Therefore, we can extract interesting conclusions that allow us to understand the keys to parameterize a query genetic optimizer from the model, which are independent from the random effects of the query and the population. Specifically, as an example, results show that all the factors considered in the model are statistically significant, meaning that they have a not negligible impact on the average cost. This observation is not trivial, since the contributions of the number of mutation operations executed per generation, for example, had not been studied before and, although it is quite clear that the use of this kind of operations is generally accepted, their impact for the query optimization problem had not been proved until now.

Acknowledgments: Special thanks to Calisto Zuzarte for his helpful comments and his support.

References


