1975

The Difficulty of Optimum Index Solution

Douglas E. Comer
Purdue University, comer@cs.purdue.edu

Report Number:
77-248

The Difficulty of Optimum Index Selection

Douglas Comer

Computer Sciences Department

Purdue University

CSD-TR 248
September 1977

The Difficulty of Optimum Index Selection

Douglas Comer

Keywords and Phrases: attribute selection, secondary index, index selection, complexity

CR categories: 3.73, 3.74, 4.33, 4.34

Abstract

Given a file on a secondary store in which each record has several attributes, it is usually advantageous to build an index mechanism to decrease the cost of conducting transactions to the file. The problem of selecting attributes over which to index has been studied in the context of various storage structures and access assumptions. One algorithm to make an optimum index selection requires 2^k steps in the worst case, where k is the number of attributes in the file. We examine the question of whether a more efficient algorithm might exist and show that even under a simple cost criterion the problem is computationally difficult in a precise sense. Our results extend directly to other related problems where the cost of the index depends on fixed values which are assigned to each attribute. Some practical implications are discussed.

1. Introduction:

For a file on a secondary store in which each record has several attributes, it is usually advantageous to build an index mechanism to decrease the cost of conducting transactions to the file. The problem is to determine which attributes to include in the index.

Any solution to the index selection problem must consider the file organization, the transactions conducted with the database, the cost of index creation and maintenance, and the potential value of an index in decreasing access costs. The problem has, therefore, been studied in a wide variety of contexts. Lum and Long [8] give an impirical evaluation of index selection, while others [6,9,10,11] provide a model for analysis purposes.

An approach taken in [2] is to provide an independent index for each attribute in the file. But the time and space required to maintain and update the separate indices may not be worthwhile. One alternative is to combine all attributes into one and use a single index. This is advocated in [7]. On the other hand, it was demonstrated in [3] that a complete knowledge of all queries to the database could lead to an optimum index.

We assume that the attributes of the file are not to be combined. Furthermore, we assume that although the complete set of queries is not known in advance, statistical properties can be obtained. These are a reasonable assumptions for most situations since it is usually possible to collect statistics about queries automatically even if the exact set of transactions to the database are not known.

Of particular interest to us is [10] which gives a model for the optimum index selection problem and provides an algorithm for solving

the problem under similar assumptions. Two parameters are calculated for each attribute from which the algorithm makes an index selection. It is suggested that the algorithm be used in the following manner. Since the users' requests may change over time, the system keeps statistics about the recent transactions to the file. Periodically, an optimum set of indices for the file is computed using the algorithm and the system then discards those indices which are no longer cost effective and keeps (or creates) those which are. Moreover, it is suggested that this process be repeated only after some fixed time because the algorithm requires substantial running time. The worst case running time is, in fact, exponential in the size of the input (although it is much less on the average).

We examine the question of whether a faster algorithm might be possible. Unfortunately, the question is answered in the negative. The result and some consequences are given in section 3, following precise definitions of a file and the index selection problem.

2. Definitions:

We will assume the relational model of data [4] and consider the case where there is a single relation in the database. Our results extend trivially to multi-relational systems. A file is defined, consistent with [10], as follows:

Let $A_1, A_2, \ldots, A_k$ be finite sets of attributes. A file F consists of n records $r = (v_1, v_2, \ldots, v_k)$ where each $v_i \in A_i$, the i^{th} attribute. Thus, $F \subseteq A_1 \times A_2 \times \ldots \times A_k$. In a given file not all elements of an attribute may be present. The value set of the i^{th} attribute for a file F is $V_i = \bigcup_{r \in F} v_i$. Note that $V_i \subseteq A_i$, $1 \leq i \leq k$.

The degree of a file F is given by $\max\{|V_1|, |V_2|, \ldots, |V_k|\}$, where $|V_i|$ represents the number of elements in value set V_i. Files with degree 2 will be referred to as binary files. The basic notion is that if a file has degree p then there is a file in which no entry is greater than p for which the index selection problem, as defined below, is equivalent.

The following ideas are used in the definition of the index selection problem. Attribute i is said to distinguish two records, r and s, iff they differ in the i^{th} component (i.e. v_i in r differs from v_i in s). I is an indexing set for a file F with k attributes iff $I \subseteq \{1, 2, \ldots, k\}$ and any pair of records in F is distinguished by some attribute in I. The size of an indexing set is the number of elements in it.

We think of a file as a 2-dimensional table in which rows correspond to records and columns correspond to attributes. An indexing set is a subset of the columns such that no two rows of the table have identical values for every attribute in the subset.

The Optimum Index Selection Problem (OISP) is defined as:

Given: A file F with n records and k attributes, and an integer p.

Question: Does there exist an indexing set for F with size no more than p?

OISP is stated in this simplistic form because we are interested in a proof of its difficulty. Later we will show how the results extend to seemingly more complicated problems which arise in practice.

3. Main Result:

In this section we show that OISP is difficult to solve computationally and show how this result extends to the kinds of problems that occur in practice.

THEOREM 1: OISP is NP-Complete[1] for files of degree d, d $\geq$ 2.

PROOF: The details of an NP-Completeness reduction are given in Appendix A. □

This theorem says that OISP is in a large class of combinatorial problems which are known to be difficult. The class of NP-Complete problems includes well-known problems such as the traveling salesperson problem and the bin packing problem. Although there is no proof that an NP-Complete problem is inherently difficult, no algorithm has been found for any problem in this class which has less than exponential running time for arbitrary inputs. Furthermore, finding an efficient algorithm for any problem in the class would be tantamount to finding an efficient algorithm for all NP-Complete problems. Thus, one should assume that a program to solve OISP on an input file of k attributes might require as many as 2^k steps (or worse). And running the same program on a file with k + 1 attributes might take twice as long! So the program will only be practical for small values of k (if it is practical at all).

Observe that we have selected a rudimentary problem and shown that any program to solve it will be inefficient. Since this simple problem is difficult, it follows that more complicated forms of the problem would also require large amounts of computer time to solve. Furthermore, the result is strong in that it applies even if the attribute values are restricted to the binary range. To see how this result extends to the case where the indexing set selection is also based on a value function, consider a Modified Index Selection Problem (MISP) which is a simplification of the one given in [10]. Let the probability of access

---

[1] The reader is referred to Aho et al [1] for details of NP-Complete problems. It is reasonable to substitute "computationally difficult" in place of "NP-Complete".

of attribute A_i be given by p_i (subject to $\sum_{i=1}^{k} p_i = 1$). Let the access value of an indexing set I be the sum over all attributes in I of p_i. It is desirable to choose an indexing set with highest access value; and yet, one would not like to index over every attribute in the file. One compromise might be to select a minimum size indexing set which had the highest access value. Let MISP be the problem of finding a minimum size indexing set of maximum access value.

THEOREM 2: MISP is at least as difficult as OISP.

PROOF: Suppose that there were an efficient program, say P, to solve MISP. We could use P to solve OISP as follows. Let $p_i = 1/k$, $1 \leq i \leq k$. A minimum size indexing set would be produced by P efficiently. From the size of the set OISP could easily be answered. But this is a contradiction: we know that OISP is difficult to solve, so program P could not exist. If no efficient program for MISP exists, then MISP is as difficult as OISP. □

In essence we have argued that if MISP were easy, OSIP would be easy. Similar arguments apply to other problems in index selection.

4. Consequences and Conclusions:

We have shown that the index selection problem can be difficult even for simple cost criterion using only binary files. It follows that more complicated criteria only serve to make the problem harder. This result, then, is a warning: although the computation time to find an optimum index may be tolerable for some files, there are cases for which it will be exponential.

We conclude that: 1) any program to solve the index selection problem may require large amounts of computer time (on some inputs), 2) adding even 1 attribute to a file could double the running time of such a program, 3) it would be unwise to incorporate such an algorithm in a database

system in which optimum indices were recomputed after every update (or every few updates), and 4) the algorithm given in [10] will probably not be improved.

Looking at our result in a different way, one can see that any "efficient" program to solve the index selection problem (one which requires only a polynomial amount of running time for any input) cannot always choose an optimum set of attributes. In a sense, any fast program must be incorrect, at least some of the time.

Despite the fact that an optimum indexing set is difficult to find, it may be easy to approximate a solution quickly. In fact, this result motivates the study and analysis of efficient approximation algorithms. In situations where an approximation algorithm produced a good (but not optimum) index selection, it could be used more often to keep the file close to optimum. Over the long run such a solution could prove to be quite beneficial. Therefore, more work in this area is encouraged.

Appendix A

(reduction for Theorem 1)

We will reduce SAT3 (satisfiability with exactly 3 literals per clause) to OISP. Karp [3] shows that SAT3 is NP-Complte.

Let an instance of SAT3 be a Boolean formula in conjunctive normal form with exactly 3 literals in each of its m clauses. Let there be 2n literals in B (denoted x, $\bar{x}$, y, $\bar{y}$, z, $\bar{z}$ in the construction). Construct a file of 2m + n + 1 records and m + 2n attributes as shown in Figure 1. We claim that there is an indexing set for F of size n + m iff B is satisfiable.

Suppose B is satisfiable. Let H be a set of n literals which satisfy B such that no pair of complementary literals appears in H. Form an indexing set as follows: select all m attributes from set P (as shown in Figure 1) and n attributes from set Q which correspond to literals in H. Clearly the records in set K are distinguished by the n selections from Q. Furthermore, records in set J are divided into pairs by the selections from set P. Since H satisfies B, it must be that for each pair of records in J there is some attribute in Q which corresponds to a literal in H that is selected and hence distinguishes the two records in the pair. Thus, if B is satisfiable, n + m attributes are sufficient to distinguish all records.

Now suppose that there is an indexing set, I, of size n + m. At least m attributes from set P must be in I or records 2, 4, ..., 2m could not be distinguished from the last record. Similarly, at least n attributes from set Q must be included in I, one from each pair of attributes corresponding to complementary literals, or records in set K could not be distinguished from the last record. But consider the

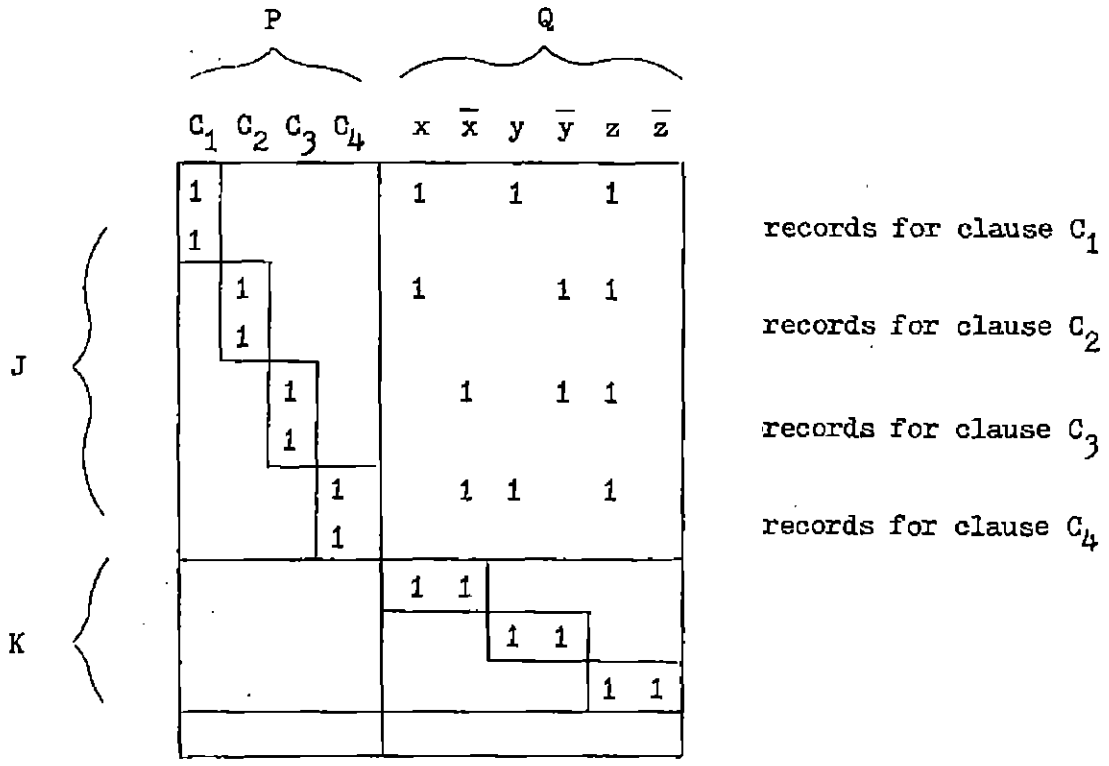

Figure 1: Sample construction for $B = C_1 \cdot C_2 \cdot C_3 \cdot C_4$, where $C_1 = (x+y+z)$, $C_2 = (x+\bar{y}+z)$, $C_3 = (\bar{x}+\bar{y}+z)$, and $C_4 = (\bar{x}+y+z)$. There are m attributes in set P, one for each clause of B, and 2n attributes in Q which correspond to the 2n literals of B. All values not shown are zero.

pairs of records formed by selections of attributes in set P. It must be that for each pair at least one attribute was selected from Q which distinguished the pair. Let H be the set of n literals in B which correspond to the n selections made from set Q. From the construction we have that $H \cap C_i \neq \emptyset$, for $1 \leq i \leq m$. Thus, H satisfies B.

Since OISP can be solved on a nondeterministic Turing Machine in polynomial time, the theorem follows.

References

[1] Aho, A., Hopcroft, J., and Ullman, J., The Design and Analysis of Computer Algorithms, Addison Wesley, 1974.

[2] Blier, R. and Vorkaus, A., "File Organization in the SCD Time Shared Data Management (TDMS) System," Proc. of the 1968 IFIP Congress.

[3] Casey, R., "Design of Tree Structures for Efficient Querying," Comm. of the ACM, vol 16:9 (Sept 73), pp. 549-556.

[4] Codd, E., "A Relational Model of Data for Large Shared Data Banks," Comm. of the ACM, vol 13:6 (June 70), pp. 377-387.

[5] Karp, R., "Reducibility Among Combinatorial Problems," in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher eds., Plenum Press, N.Y., 1972, pp. 85-103.

[6] King, W., "On the Selection of Indices for a File," IBM research report RJ 1341, San Jose, CA, January 1974.

[7] Lum, V. and Long, H., "Multi-Attribute Retrieval with Combined Indexes," Comm. of the ACM, vol 13:11 (Nov 70), pp. 660-665.

[8] Lum, V. and Long, H., "An Optimization Problem on the Selection of Secondary Keys," Proc. ACM National Annual Conference, 1971.

[9] Palermo, F., "A Quantative Approach to the Selection of Secondary Indexes," IBM research report RJ 730, San Jose, CA, July 1970.

[10] Schkolnick, M., "The Optimal Selection of Secondary Indices for Files," Information Systems, vol 1:4 (1975), pp. 141-146.

[11] Stonebraker, M., "The Choice of Partial Inversions and Combined Indices," International Journal of Computer and Information Science, vol 3:2 (June 74), pp. 167-188.