1974

Tree Structures Construction Using Hit Ratios

S. B. Yao

Report Number:
75-133
TREE STRUCTURES CONSTRUCTION USING HIT RATIOS

S. B. Yao
Department of Computer Science
Purdue University
Lafayette, Indiana 47907

CSD-TR-133
November 1974
1. Introduction

Many prefix (digital) tree structures are available for organizing files; each has advantages and disadvantages. The choice among the tree structures depends on the particular design objectives. In this paper, a general model of tree structure is proposed. The model parameters are determined by analyzing the access activities of the identifier set.

The hit ratio of an operation on a set of elements is the relative proportion of elements selected or accessed by the particular operation performed. In the analysis of applications in data processing, the hit ratio is frequently an important parameter affecting the searching of a file. In particular, when the set is partitioned into groups, the group hit ratio is the proportion of groups with at least one element hit. It is often desirable to find the relationship between the hit ratio and the group hit ratio. One of the places where this type of problems is encountered is the analysis or simulation of file organizations in data bases. Previously, various hit ratios were at best approximated [2]. A solution to this problem was attempted by Severance [5]. The results obtained are computationally intractable, and hence not practical. In this paper, a simple closed expression for computing the group hit ratio is obtained.

In order to demonstrate the use of hit ratios, an application to the determination of general tree structure is presented. The basic tree structures considered are multi-dimensional index, TRIE and (doubly chained) TREE [3,4,1]. It has been suggested by Sussenguth [7] that these structures should be used for high, medium and low group hit ratios (filial densities) respectively. From the observation that the group hit ratio is high near the root and is low near the terminal, a simple two-level hybrid structure TRIE-TREE...
was introduced by Severance [6]. It in fact uses a multi-dimensional index on the first level whose active pointers point to labeled index tables on the second level. The TRIE-TREE was shown to minimize the storage required if the group hit ratio can be approximated by a step function - unity from root to level \( k \) and minimum from level \( k + 1 \) to terminals. In the following sections, the expressions for computing the group hit ratios are given, and the algorithm for constructing a general hybrid structure, the indexed tree, is introduced.

II. The Hit Ratio Theorem

Theorem 1. Given \( n \) elements grouped into \( m \) buckets \((1 < m \leq n)\), each contains \( n/m \) elements. If \( r \) elements \((r \leq n - n/m)\) are randomly selected from the \( n \) elements, the expected number of buckets hit (buckets with at least one element selected) is given by

\[
m \cdot \left( 1 - \prod_{i=1}^{r} \frac{nd - i + 1}{n - i + 1} \right) \quad \text{where} \quad d = 1 - \frac{1}{m} \quad (1)
\]

Proof: Let \( X \) be a random variable representing the number of buckets hit and let \( I_k \) be a random variable where

\[
I_k = \begin{cases} 
1 & \text{when at least one element in the } k\text{-th bucket is selected} \\
0 & \text{otherwise}
\end{cases}
\]
The k-th bucket has \( p = n/m \) elements and there are \( n - p \) elements not in the k-th bucket. The probability that no elements are selected from the k-th bucket is

\[
\frac{\binom{n-p}{r}}{\binom{n}{r}} \text{ or } \frac{\binom{nd}{r}}{\binom{n}{r}} \quad \text{where } d = 1 - \frac{1}{m}.
\]

It follows that the expectation of \( I_k \) is

\[
E(I_k) = 1 - \frac{\binom{nd}{r}}{\binom{n}{r}}.
\]

Hence the expected number of buckets hit is

\[
E(X) = \sum_{k=1}^{m} E(I_k) = m \left( 1 - \frac{\binom{nd}{r}}{\binom{n}{r}} \right)
\]

\[
= m \left( 1 - \frac{(nd)!(n-r)!}{n!(nd-r)!} \right)
\]

\[
= m \left( 1 - \prod_{i=1}^{r} \frac{nd - i + 1}{n - i + 1} \right)
\]

Q.E.D.

The following corollary of the theorem is obvious:

**Corollary 1.** If \( r > n - n/m \) or \( m = 1 \), then all \( m \) buckets are hit.
The following corollary provides a simpler approximation of the theorem.

**Corollary 2.** If the \( r \) elements are sequentially selected from the \( n \) elements grouped into \( m \) buckets and each element is randomly selected from the \( n \) elements (i.e. selection with replacement), the expected number of buckets hit is given by

\[
m \cdot (1 - (1 - \frac{1}{m})^r)
\]

and is independent of the total number of elements \( n \).

**Proof:** The probability for a bucket to be hit is

\[
1 - (1 - \frac{1}{m})^r
\]

This is true for any of the \( m \) buckets. It follows that the expected number of buckets hit is

\[
m \cdot (1 - (1 - \frac{1}{m})^r)
\]

Q.E.D.

The approximation of this corollary is good if \( r \ll n \) and \( m \ll n \), since for large \( n \) expression (1) approaches expression (2). Intuitively, when \( n \) is large and each group contains many elements, the selection with and without replacement makes little difference. This is not true, however, for moderate \( r \) and \( n \). A comparison between the two methods is plotted in Figure 1.
Figure 1. Comparison of selection with and without replacement.
When records are allocated in secondary devices, they are usually grouped into buckets (blocks or pages). It is often desirable to compute the expected number of buckets to be accessed when records are retrieved. The approximated value in Corollary 2 are frequently used for this purpose [2]. A more precise evaluation should be based on the result of the above theorem.

III. Filial Set Density

An interesting application of the hit ratio theorem is in the determination of general tree structure. In this discussion, the terminology defined in Sussenguth is employed with some modification [7]. A tree is a graph which contains no circuits and has at most one branch entering each node. A root is a node which has no branches entering it and a terminal is a node which has no branches leaving it. The node \( x \) is a parent node of the node \( y \) if there is a branch from \( x \) to \( y \). The filial set of a node \( x \) is the set of all nodes of which \( x \) is the parent. The set of nodes which lies at the end of all paths of length \( i \) from the root of a tree comprises the \( i \)-th level of the tree. The size of a tree at a level is the sum of all filial set sizes at that level. The degree of a tree is the maximum number of branches leaving a node. A \( p \)-ary tree is a tree of degree \( p \).

Terminal nodes correspond to the records stored and internal nodes to components of the identifiers (keys) of the records. Each identifier can be partitioned into segments. A prefix tree
is a tree that decodes one segment at each level. For a set of equal-length identifiers, a prefix tree has all terminal nodes on the same level. In this analysis, the pieces of information stored in the records are immaterial. The properties of active identifiers associated with the records stored are studied.

Assume that there are \( n \) possible identifiers and among them \( r \) are active identifiers. The identifier density is given by \( D = r/n \). Assume further that the identifier set is structured by an \( m \) level \( p \)-ary balanced prefixed tree. From the definition we have \( p^m = n \) and the maximum possible size of the tree at level \( i \) is \( s_i = p^i \). It follows that the possible tree size on level \( i \) is \( s_i = n^{i/m} \) and is independent of the degree \( p \) of the tree.

To compute the active size of the tree, assume that the \( r \) active identifiers are randomly distributed among the \( n \) possible identifiers. Figure 2 shows that the active tree size at level \( i \) is the number of nodes on level \( i \) that has at least one active identifier in its subtree. This can be viewed as randomly selecting \( r \) elements from the \( n \) elements grouped into \( n^{i/m} \) buckets, each containing \( n^{1-i/m} \) elements. Using the hit ratio theorem, the expected active tree size \( A_i \) at level \( i \) is found:

\[
A_i = n^{i/m} \left(1 - \prod_{j=1}^{r} \frac{nd - j + 1}{n - j + 1}\right) \quad \text{where} \quad d = 1 - \frac{1}{n^{1/m}} \quad (3)
\]
Figure 2 Sizes of an $m$ level prefix tree

$S_1 = n^{i/m}$

$r$ active identifiers

$n$ possible identifiers
The values of $S_i$ and $A_i$ are plotted in Figure 3 for the case of $D = 0.1$ and $n = 100,000$.

There are $A_{i-1}$ filial sets at level $i$ of the tree. The average filial set size at level $i$ is obtained by dividing the adjacent active tree sizes:

$$W_i = \frac{A_i}{A_i - 1} \quad (4)$$

Figure 4 shows the average filial set sizes for three different identifier densities. If we define the filial set density of level $i$ by $D_i = W_i/p$, Figure 4 shows that the filial sets are very dense near the root of the tree and become sparse towards the terminals of the tree. In the following section, the filial set densities are used to determine the design of tree structure.

IV. TRIE and TREE

The way that a segment is decoded in a level depends on the structure one chooses to organize that level. Within any given level, two alternative solutions, TRIE and TREE, are suggested by Fredkin [4] and by de la Briandais [1]. The difference between the two techniques is illustrated in Figure 5 which shows the filial set structures of one identifier segment. The maximum number of different symbols that an identifier segment can represent is $p$. When a segment is a single character, $p$ is the vocabulary size.
n = 100,000
r = 10,000
m = Total number of levels

Figure 3 Possible and active tree sizes
Figure 4. Expected filial set sizes
TRIE - without label, for high active filial set

TREE - with label, for low active filial set

Figure 5. Filial set structures
The TRIE represents filial sets in level \( i \) by arrays of \( p \) pointer fields which contain pointers to the next level. Each active pointer field lies on the \( i \)-th level of an access path. For those pointer fields without an access path, a null pointer is stored. Since each of the \( p \) possible symbols corresponds to a field in the array, the search needs only one comparison. Let \( V \) denote the length of a pointer. The storage requirement of one TRIE filial set is

\[
S_1 = p \cdot V
\]  

(5)

and is independent of the density of the filial set (Figure 6). The storage overhead for null pointers makes TRIE filial set only suitable for relatively high filial density.

The TREE represents filial sets in level \( i \) by storing only the pointers for the active symbols of segment \( i \). The identifier segment stored with the filial set is a label of length \( L \). The label and pointer pairs are assumed to be stored sequentially. Assuming a uniform distribution of active symbols, a search of one half of the elements in a filial set is expected for decoding an identifier segment. The number of comparisons is \( \frac{p \cdot D_i}{2} \). The storage requirement of one TREE filial set is

\[
S_2 = p \cdot D_i \cdot (L + V)
\]  

(6)

The storage overhead for labels makes TREE filial set only suitable for relatively low filial density.
Figure 6. Storage requirement of a filial set
Let the cost of unit storage be $C_s$ and the cost of one comparison be $C_t$. From the above derivation, the level $i$ cost of Trie is

$$C_{TRIE} = C_s \cdot p \cdot V \cdot A_{i-1} + C_t$$

(7)

and the level $i$ cost of Tree is

$$C_{TREE} = C_s \cdot (L + V) \cdot A_i + C_t \cdot \frac{p \cdot D_i}{2}$$

(8)

where $D_i$ is the identifier density of the $i$th segment of an identifier.

**Theorem 2.** The condition for the first Tree level $i$ to be preferred is

$$D_i < \frac{C_s \cdot p \cdot V \cdot A_{i-1} \cdot V + C_t}{C_s \cdot p \cdot V \cdot A_{i-1} \cdot (L + V) + C_t \cdot (p/2)}$$

(9)

**Proof.** The condition results from $C_{TRIE} > C_{TRIE}$. From expressions (7) and (8), we obtain $C_s \cdot p \cdot V \cdot A_{i-1} + C_t > C_s \cdot (L + V) \cdot A_i + C_t \cdot \frac{p \cdot D_i}{2}$.

Hence (9) follows immediately by substitution $A_i = D_i \cdot A_{i-1}$.

Q.E.D.

Theorem 2 reveals that by examining the filial set density and the active tree size, a break point for Trie and Tree levels can be determined.

**V. Indexed Tree**

Multi-dimensional index [3] permits rapid access for files with extremely high density. The $i$-th dimension can be viewed as
the i-th level of a tree. It is shown by Sussenguth [7] and
Severence [6] that when the filial density is very high, \( D_i = 1 \),
many TRIE levels should be combined into a multi-dimension index.
The storage requirement of an m-dimensional index is \( p^m \cdot v \), as
compared to the storage requirement of an m-level TRIE which is
\( \frac{p \cdot (p^m - 1) \cdot v}{p - 1} \). The access to multi-dimensional index is also faster
since only one comparison is needed instead of \( m \).

It is interesting to see how many TRIE levels can be replaced
by this technique. Assume that the first \( i - 1 \) levels, \( i \geq 1 \),
were assigned a multi-dimensional index and consider the i-th
level structure. If TRIE structure is chosen for level \( i \), the
additional cost, denoted by \( F_1 \), is the cost for a single level
TRIE which includes the storage cost of active filial sets and
TRIE search time.

\[
F_1 = C_s \cdot p \cdot V \cdot A_{i-1} + C_t
\]  \hspace{1cm} (10)

If this level is included in the multi-dimensional index the
additional costs are

\[
F_2 = C_s \cdot p^i \cdot V + C_t \\
i = 1
\]  \hspace{1cm} (11)

\[
F_2 = C_s \cdot p^i \cdot V + C_t - (C_s \cdot p^{i-1} \cdot V + C_t)
\]  \hspace{1cm} (12)

\[
= C_s \cdot p^{i-1} \cdot (p - 1) \cdot V \\
i > 1
\]
Note that there is no additional cost for search time when $i > 1$ since it is independent of the size of the index.

**Theorem 3.** The $i$-th level of a TRIE is to be combined into the index, whenever

$$A_{i-1} \geq p^{i-2} \cdot (p - 1) - \frac{C_t}{C_s \cdot p \cdot v} \quad \text{for all } i > 1 \quad (13)$$

**Proof.** The result comes from the condition that $F_1 \geq F_2$ and expressions (10) and (12).

Q.E.D.

**Corollary 3.** When $i = 1$, there is no difference between an index and a one-level TRIE.

The above analysis shows that a tree structure should have the general form of an index followed by TRIE and TREE structures. Such an indexed tree is illustrated in Figure 7.

**VI. Conclusion.**

An important group hit ratio problem is identified and a closed expression is given for its solution. The results can be applied to many data base analysis problems where the number of pages accessed is to be estimated. The simplicity of the results makes it applicable for a range of similar problems. A number of applications to the evaluation of file organizations was shown by Yao [8].
Figure 7. An indexed tree
The hit ratio theorem is applied to solution of a tree structuring problem. Many tree structuring techniques are available and each has different characteristics. By investigating the identifier densities on various levels of a tree, a hybrid tree structure is developed. The hybrid indexed tree, employing tree structuring techniques at levels where they are most efficient, provides a new and more efficient technique for file organization.
REFERENCES


