Expressions for Completely and Partly Unsuccessful Batched Search of Sequential and Tree-Structured Files

YANNIS MANOLOPOULOS AND J. (YANNIS) G. KOLLIAS

Abstract—**A number of previous studies derived expressions for batched searching of sequential and tree-structured files on the assumption that all the keys in the batch exist in the file, i.e., all the searches are successful. New formulas for batched searching of sequential and tree-structured files are derived, but the assumption made now is that either all or part of the keys in the batch do not exist in the file, i.e., the batched search is completely or partly unsuccessful.**

Index Terms—**Access strategy, batched searching, performance evaluation, physical database design, sequential and tree-structured files, successful and unsuccessful search.**

I. Introduction

THIS paper considers a file residing in a secondary storage device, which is physically partitioned into fixed size blocks (e.g., disks). A query based on a primary key value (e.g., social security number) is satisfied by one record (i.e., successful search) or the requested record does not exist in the file (i.e., unsuccessful search). The studies in [1], [2] present a number of file organization schemes (e.g., sequential, random, tree-structured, etc.) and estimate the cost of both successful and unsuccessful searches using these schemes. These costs are expressed by the required number of block accesses to satisfy the query. A query based on secondary key values (e.g., date of birth, sex, etc.) is satisfied by accessing a number of records located normally in more than one block of secondary memory. The blocks containing the records of interest are usually established by employing secondary indexing techniques [1], [2].

Let us assume that we have to satisfy k queries based on either primary or secondary key values. Shneiderman and Goodman [3] argued that the response time of satisfying the queries may be reduced if we consider them as a **batch** instead of satisfying them individually on a first-come-first-served basis. A number of studies considering batching appeared in the literature. Mainly, they report estimations on the number of blocks of secondary storage that have to be transferred in main memory for various environments. The assumptions made by all previous studies is that the records are retrieved under a replacement or a nonreplacement model. The replacement (nonreplacement) model assumes that the probability of locating a record in a specific block is not (is) reduced when a block has already been accessed.

Studies based on the replacement model assumption derived expressions for the expected value of block accesses required to satisfy a request for k keys using sequential [4] or random files [4]–[6]. Under the assumption of the nonreplacement model, expressions have also been derived for sequential [3], [4], [7]–[9], random [4], [10]–[13] and tree-structured files [3], [8], [14]. Table I lists the above mentioned studies according to the file organization and the model concerned.

In this paper we focus on batched searching of sequential and tree-structured files and, therefore, we start discussing the relative studies in more detail. In [3] approximate formulas are derived evaluating the gain due to batched searching of sequential files and of j-ary search trees. In [7] another approximate solution was given for the cost of batched searching in sequential file structures. Recently, [8] derived exact and approximate formulas for the cost of batched searching in both the sequential and tree-structured environments. The same problem for tree-structured files was also examined in [14], where an accurate formula for the gain was derived. We note that similar exact formulas were derived estimating the cost of batched searching in an array [15] and in a main memory database [16], as well as the cost of seeking in a disk system [17]–[19].

A common characteristic of all the previous studies is that they assume that all the records of the batch exist in the file, i.e., that the search is successful. In this paper the last assumption is dropped and the performance of completely or partly unsuccessful batched searching is examined. We say that a batched search is **completely unsuccessful** or **partly unsuccessful** when all the keys or some keys of the batch do not exist in the file respectively. Before proceeding further, we note that erroneous input and missing records from the file (possibly because file updates are performed off-line) are among the reasons which may cause completely and partly unsuccessful

Manuscript received August 31, 1987; revised September 30, 1988.
Y. Manolopoulos is with the Department of Electrical Engineering, Division of Electronics and Computer Engineering, Aristotelian University of Thessaloniki, 54006 Thessaloniki, Greece.
J. G. Kollias is with the Department of Electrical Engineering, Division of Computer Science, National Technical University of Athens, 15773 Zografou, Athens, Greece.
IEEE Log Number 8927390.

0098-5589/89/0600-0794$01.00

TABLE I
REFERENCES TO PREVIOUS STUDIES

		MODEL	
		Replacement	Non-Replacement
FILE STRUCTURE	sequential	[4]	[3,4,7-9]
	random	[4-6]	[4,10-13]
	tree-structured		[3,8,14]

batched searches. As it was mentioned above, the case of an unsuccessful search is always considered when evaluating the performance of single queries to a file [1], [2].

In the next two sections, expressions are derived for completely and partly unsuccessful batched search of sequential files and j-ary trees. Initially, we consider the extreme case where no record in the file matches any of the keys in the batch. The study proceeds to the partly unsuccessful batched search where some keys of the batch exist in the file and some do not exist. The last section presents some numerical examples and draws the conclusions.

II. SEQUENTIAL FILE

As derived in [8], the cost of successful batched search, in terms of the expected value of block accesses, is:

$$\mathrm{COST}_{\mathrm{suc}} = (n+1)k/(k+1) \tag{1}$$

where n is the number of the file records (occupying one block each) and k is the size of the batch.

Suppose that a sequential file consists of n unsorted records, occupying one block each, and the batch consists of k records, either sorted or unsorted. If at least one of the records in the batch does not exist in the file, then exactly n block accesses have to be made.

Consider now the case that both the file records and the batch records are sorted in the same order. In this case, an unsuccessful search is detected whenever the value of the key record is greater than the key value of the batch record. After the detection of an unsuccessful search, the searching resumes from the last examined file record. The following subcases must be examined.

A. Completely Unsuccessful Batched Search Under a Nonreplacement Model

In between the n file records $n + 1$ subintervals are created. For the moment, we assume that the k records retrieved obey a nonreplacement model, which means that any two batched records belong to different subintervals. This case may arise when the k records are distinct. Using an analysis similar to that of [16], we derive that the cost of the completely batched unsuccessful search under the nonreplacement model, in terms of the expected value of block accesses, is as follows:

$$\mathrm{COST}_{\mathrm{uns,nrep}} = \frac{1}{C(n+1,k)} \cdot \left[\sum_{i=k}^{n} iC(i-1,k-1) + nC(n,k-1)\right].$$

This expression is explained as follows. The cost of this completely unsuccessful batched search is equal to the cost of searching for the last record of the batch. Since the nonreplacement model is assumed, this last record may not lie before the kth subinterval. The probabilities that this record may lie in the kth up to the $(n+1)$th subinterval are assumed to be equal. The sum in the parenthesis gives the cost (in block accesses) of the unsuccessful searches in each of the first n subintervals, times the number of ways the rest $(k-1)$ records of the batch may lie in the rest $(i-1)$ subintervals. In an analogous manner, the second term of the parenthesis gives the cost of an unsuccessful search in the last subinterval, times the number of ways the remaining records of the batch may lie in the first n subintervals. It is worth noting that the cost of this last search is n and not $(n+1)$ block accesses. The quantity of the parenthesis is divided by the number of the ways the k records may lie in the $(n+1)$ subintervals. Therefore,

$$\begin{aligned}\mathrm{COST}_{\mathrm{uns,nrep}} &= \frac{1}{C(n+1,k)} \cdot \left[\sum_{i=k}^{n+1} iC(i-1,k-1) - C(n,k-1)\right] \\ &= (n+2)\frac{k}{k+1} - \frac{C(n,k-1)}{C(n+1,k)} \\ &= (n+2)\frac{k}{k+1} - \frac{k}{n+1} \end{aligned}\tag{2}$$

where $C(a, b)$ is the a-choose-b combination.

B. Completely Unsuccessful Batched Search Under a Replacement Model

Now, we assume that the k records obey a replacement model. This means that a number of batched records may be retrieved from the same subinterval, i.e., the batch may contain nondistinct records. The analysis for the cost of the completely unsuccessful batched search under the replacement model is similar to that of the previous section. The only difference is in the estimation of the number of ways the records of the batch may lie in the possible subintervals. This is depicted in the bounds of the summation and the combinations. Finally, in terms of the expected value of block accesses, this cost is

$$\begin{aligned}\mathrm{COST}_{\mathrm{uns,rep}} &= \frac{1}{C(n+k,k)}\left[\sum_{i=1}^{n} iC(i+k-2,i-1) + nC(n+k-1,n)\right]\\ &= \frac{1}{C(n+k,k)}\left[kC(n+k,k+1) - (k-1)C(n+k-1,k) + nC(n+k-1,n)\right]\\ &= n\frac{k}{k+1} + \frac{n}{n+k} \quad \text{for } k \geq 1. \qquad (3)\end{aligned}$$

C. Partly Unsuccessful Batched Search

In this case of partly unsuccessful batched search many possibilities arise, because existing and nonexisting records may obey a replacement and a nonreplacement model. We proceed to the analysis according to the two assumptions that the k_1 (k_2) records retrieved obey a nonreplacement (replacement) model.

Having in mind that the last record in the batch may be either existing or nonexisting in the file, the analysis is based on that of the previous subsections. The cost of a partly unsuccessful batched search, in terms of the expected value of block accesses is:

$$\mathrm{COST}_{\mathrm{uns,par}} = \frac{1}{C(n,k_1)\,C(n+k_2,k_2)} \cdot \left[\sum_{i=k_1}^{n} i\big(C(i+k_2-1,k_2)C(i-1,k_1-1) + C(i+k_2-2,k_2-1)\,C(i-1,k_1)\big) + nC(n,k_1)\,C(n+k_2-1,k_2-1)\right].$$

The denominator gives the number of ways that the k_1 and k_2 records may lie in the n and $(n+1)$ subintervals, respectively. The second term in the parenthesis represents the combined cost of the successful and the unsuccessful search of the last record of the batch in case it corresponds to the last record and the last subinterval of the file, respectively, times the number of ways this case may happen. The summation of the parenthesis concerns all the remaining cases that the last record of the batch may (not) match any file record (subinterval) but the last one. After some algebra based on the properties of combinations and the binomial coefficient relations [1], we can derive the following relation:

$$\begin{aligned}\mathrm{COST}_{\mathrm{uns,par}} &= \frac{n^2(k_1+k_2)^2 + nk_1(k_1+k_2) + k_2(n-k_1)}{(n+k_2)(k_1+k_2)(k_1+k_2+1)}\\ &\quad + \frac{nk_2}{n+k_2}. \qquad (4)\end{aligned}$$

Relations (1) and (3) are produced from (4) by assigning $k_2 = 0$ and $k_1 = 0$, respectively.

III. J-Ary Search Tree

A j-ary search tree [3], [8], [14] is a B-tree [1], [2] having at maximum j sons per father node or, equivalently, at maximum $j-1$ records in every node. The tree is also characterized by its height l, i.e., the maximum distance from the root to the bottom node. Thus the tree has $l+1$ levels. For the sake of mathematical analysis, it is assumed that the tree is complete and, therefore, the number of nodes is:

$$1 + j + j^2 + \cdots + j^l = (j^{l+1} - 1)/(j-1).$$

The number of records in the tree is $j^{l+1} - 1$ and the number of subintervals between the records is j^{l+1}.

We start our analysis by introducing the analysis in [8]. By assuming a nonreplacement model it was proved that the cost of batched search of k records in a j-ary tree with $l+1$ levels is:

$$\mathrm{COST}_{\mathrm{suc}}(k, l+1) = 1 + j\sum_{i=1}^{k} \mathrm{PROB}_{\mathrm{suc}}(i,l,k)\,\mathrm{COST}_{\mathrm{suc}}(i,l) \qquad (5)$$

where

$$\mathrm{PROB}_{\mathrm{suc}}(i,l,k) = C(j^l - 1, i)\,C(j^{l+1} - j^l, k-1)/C(j^{l+1} - 1, k).$$

The initial conditions for $\mathrm{COST}_{\mathrm{suc}}(i,l)$ are set as follows:

$\mathrm{COST}_{\mathrm{suc}}(0, l) = 0$ for all l and

$\mathrm{COST}_{\mathrm{suc}}(i, 1) = 1$ for all $i > 0$.

Some explanations are necessary about the recursive formula (5). $\mathrm{PROB}_{\mathrm{suc}}(i,l,k)$ gives the probability that i records of the batch out of the k ones exist in a subtree of height l. Therefore, the expected number of total accesses is equal to one access for the root plus the expected number of accesses in the j subtrees of the root. For any subtree out of the j ones in any level we multiply the probability that it contains i (out of the k) records by the search cost of the corresponding subtree.

A. Completely Unsuccessful Batched Tree Search

In case of an unsuccessful tree search it is certain that the bottom level will be reached. Having this in mind and under the assumption of the nonreplacement model it is easy to proceed to the analysis with a similar reasoning to that of the previous paragraphs. Therefore, the cost of batched tree search of k nonexisting records, in terms of the expected number of block accesses, is

$$\mathrm{COST}_{\mathrm{uns}}(k, l+1) = 1 + j\sum_{i=1}^{k} \mathrm{PROB}_{\mathrm{uns}}(i,l,k)\,\mathrm{COST}_{\mathrm{uns}}(i,1) \qquad (6)$$

where

$$\mathrm{PROB}_{\mathrm{uns}}(i, l, k) = C(j^l, i)\, C(j^{l+1} - j^l, k - i)/C(j^{l+1}, k).$$

The initial conditions for $\mathrm{COST}_{\mathrm{uns}}(i, l)$ are set as follows:

$$\mathrm{COST}_{\mathrm{uns}}(0, l) = 0 \text{ for all } l$$

$$\mathrm{COST}_{\mathrm{uns}}(1, l) = 1 \text{ for all } l$$

and

$$\mathrm{COST}_{\mathrm{uns}}(i, 1) = \begin{cases} 1 & \text{if } i \le j \\ 0 & \text{if } i > j. \end{cases}$$

If a replacement model is assumed, then (6) is still valid, but now

$$\mathrm{PROB}_{\mathrm{uns}}(i, l, k) = C(j^l + i - 1, i)\, C(j^{l+1} - j^l + k - i - 1, k - i)/C(j^{l+1} + k - 1, k)$$

and

$$\mathrm{COST}_{\mathrm{uns}}(i, 1) = 1 \text{ for all } i, j \text{ and } i > 0.$$

B. Partly Unsuccessful Batched Tree Search

Suppose that the batch consists of k_1 existing and k_2 nonexisting records. Again, the k_1 or k_2 records may obey either the nonreplacement or replacement model. The expected value of block accesses to perform the batched search in a j-ary tree with l levels is:

$$\mathrm{COST}_{\mathrm{par}}(k_1, k_2, l + 1) = 1 + j \sum_{i=0}^{k_1} \sum_{m=0}^{k_2} \mathrm{COST}_{\mathrm{par}}(i, m, l)\, \mathrm{PROB}(i, m, l) \quad (7)$$

where the initial conditions are defined as follows:

$$\mathrm{COST}_{\mathrm{par}}(i, 0, l) = \mathrm{COST}_{\mathrm{suc}}(i, l),$$

$$\mathrm{COST}_{\mathrm{par}}(0, m, l) = \mathrm{COST}_{\mathrm{par}}(m, l),$$

$$\mathrm{COST}_{\mathrm{par}}(i, m, 1) = 1 \text{ for } i, m > 0$$

and

$$\mathrm{PROB}(i, m, l) = \mathrm{PROB}_{\mathrm{suc}}(i, l)\, \mathrm{PROB}_{\mathrm{uns}}(m, l).$$

The explanation of these relations is obvious. The model obeyed by the records of the batch may be monitored through the probability distributions $\mathrm{PROB}_{\mathrm{suc}}(i, l)$ and $\mathrm{PROB}_{\mathrm{uns}}(m, l)$, which have been defined earlier.

C. Approximate Formulas

In this section, some simpler approximate expressions in place of the recursive ones will be derived. In [8] the probability of a tree node not being selected is estimated. This quantity is

$$Q_{\mathrm{suc}} = \left(1 - k_1/(j^{l+1} - 1)\right)^{j-1}.$$

This relation is based on a formula derived in [6], which is a very good approximation to the exact but computationally expensive one which appeared in [13]. These formulas estimate the expected value of block accesses in a random file under the nonreplacement model. By quoting from [8], the formula for Q_{suc} is explained as follows. The total number of keys in the $(l + 1)$ levels of the tree is $j^{l+1} - 1$. The k_1 records of the batch are retrieved out of the keys of the tree at random. Therefore, the probability that a specific key is selected is equal to the fraction $k_1/(j^{l+1} - 1)$. The probability of a key not being selected is one minus the fraction. Since a node contains $(j - 1)$ records the above formula follows.

With a similar manner we define the quantity Q_{uns} as

$$Q_{\mathrm{uns}} = (1 - k_2/j^{l+1})^j.$$

Evidently, this quantity depicts the probability that a tree node is not selected when the tree is searched for k_2 nonexisting records. The differences to the previous formula for Q_{suc} are: 1) that the total number of subintervals in between the key records is j^{l+1}, and 2) the number of subintervals in between the key records of a specific node is j.

By using the quantity Q_{uns}, (6) may be approximated by

$$1 + j(1 - Q_{\mathrm{uns}}^{j^{l-1}}) + j^2(1 - Q_{\mathrm{uns}}^{j^{l-2}}) + \cdots + j^l(1 - Q_{\mathrm{uns}}^{j^0}).$$

Some explanations for this relation are necessary. First, the unit stands for the access of the root node. At the first level, which is the one below the tree root, j nodes reside. Every node at this level corresponds to j^l subintervals. Since every node contains j subintervals, a specific node out of these j ones will not be visited with probability $Q_{\mathrm{uns}}^{j^l/j} = Q_{\mathrm{uns}}^{j^{l-1}}$. The probability that a node at this level will be visited is 1 minus the previous quantity. Since there are j nodes at this level it is easy to conclude to the second term of the summation. Hereafter, in the same way we continue to the second tree level up to the lth one. We continue by simplifying the previous approximation:

$$1 + j + j^2 + \cdots + j^l - (jQ_{\mathrm{uns}}^{j^{l-1}} + j^2Q_{\mathrm{uns}}^{j^{l-2}} + \cdots + j^lQ_{\mathrm{uns}}^{j^0}) = (j^{l+1} - 1)/(j - 1) - \sum_{i=0}^{l-1} j^{l-i}Q_{\mathrm{uns}}^{j^i}. \quad (8)$$

On the other hand by using Q_{suc} and Q_{uns}, (7) for the partly unsuccessful batched search may be approximated by

$$\begin{aligned} 1 &+ j(1 - Q_{\mathrm{uns}}^{j^{l-1}}) + jQ_{\mathrm{uns}}^{j^{l-1}}(1 - Q_{\mathrm{suc}}^{(j^{l}-1)/(j-1)}) \\ &+ j^2(1 - Q_{\mathrm{uns}}^{j^{l-2}}) + j^2Q_{\mathrm{uns}}^{j^{l-2}}(1 - Q_{\mathrm{suc}}^{(j^{l-1}-1)/(j-1)}) \\ &+ \cdots + j^l(1 - Q_{\mathrm{uns}}^{j^0}) + j^lQ_{\mathrm{uns}}^{j^0}(1 - Q_{\mathrm{suc}}^{j^0}) \\ &= (j^{l+1} - 1)/(j - 1) + \sum_{i=1}^{l} j^{l+1-i}Q_{\mathrm{uns}}^{j^{i-1}}\, Q_{\mathrm{suc}}^{(j^i-1)/(j-1)}. \end{aligned} \quad (9)$$

TABLE II

EXPECTED VALUES OF BLOCK ACCESSES IN A SEQUENTIAL FILE FOR VARIOUS VALUES OF n AND k

			n = 100	n = 1000	n = 10000
k	10	$COST_{suc}$	91.82	910.00	9091.82
		$COST_{uns,rep}$	91.82	910.08	9091.91
		$COST_{uns,nrep}$	92.63	910.90	9092.73
	20	$COST_{suc}$	96.19	953.33	9524.76
		$COST_{uns,rep}$	96.07	953.36	9524.81
		$COST_{uns,nrep}$	96.94	954.27	9525.71
	50	$COST_{suc}$	99.02	981.37	9804.90
		$COST_{uns,rep}$	98.71	981.34	9804.92
		$COST_{uns,nrep}$	99.50	982.30	9805.88

TABLE III

EXPECTED VALUES OF BLOCK ACCESSES IN A SEQUENTIAL FILE FOR VARIOUS VALUES OF k_1 AND k_2. THE FILE CONTAINS n = 100 RECORDS.

k_1 \ k_2	0	1	2	3	4	5	6	7	8	9	10
0	0.00	50.99	67.65	75.97	80.96	84.29	86.66	88.44	89.82	90.92	91.82
1	50.50	67.49	75.90	80.92	84.26	86.64	88.43	89.81	90.92	91.82	92.57
2	67.33	75.23	80.88	84.24	86.23	88.42	89.81	90.91	91.82	92.57	93.20
3	75.75	80.84	84.21	86.61	88.41	89.80	90.91	91.82	92.57	93.21	93.75
4	80.80	84.19	86.60	88.40	89.80	90.91	91.82	92.57	93.21	93.75	94.23
5	84.17	86.59	88.39	89.79	90.91	91.82	92.57	93.21	93.76	94.23	94.64
6	86.57	88.38	89.79	90.90	91.82	92.58	93.21	93.76	94.23	94.64	95.01
7	88.38	89.78	90.90	91.82	92.58	93.22	93.76	94.24	94.65	95.01	95.33
8	89.78	90.90	91.82	92.58	93.22	93.77	94.24	94.65	95.01	95.34	95.62
9	90.90	91.82	92.58	93.22	93.77	94.24	94.66	95.02	95.34	95.63	95.89
10	91.82	92.58	93.22	93.77	94.25	94.66	95.02	95.35	95.63	95.89	96.12

IV. EXAMPLES AND CONCLUSIONS

Exact and approximate formulas have been derived for the cases of completely and partly unsuccessful batched search in sequential and tree-structured files.

A. Sequential Files

Expressions (2), (3), and (4) are exact. Tables II and III show the expected values of block accesses for various parameters.

Formulas (1)–(3) are used to construct Table II. It is shown in the table and can be easily proved with simple algebra that:

1) The values of the completely unsuccessful batched search under the nonreplacement model are always greater than the values of the completely successful batched search:

$$COST_{uns,nrep} \geq COST_{suc} \text{ for all } k \leq n.$$

TABLE IV

EXPECTED VALUES OF BLOCK ACCESSES IN A COMPLETE TERNARY TREE-STRUCTURED FILE WITH 4 LEVELS FOR VARIOUS VALUES OF k_1 AND k_2 BY USING THE EXACT FORMULA (7). THE FILE CONTAINS n = 80 RECORDS.

k_1 \ k_2	0	1	2	3	4	5	6	7	8	9	10
0	0.00	4.00	6.49	8.60	10.44	12.05	13.50	14.80	16.00	17.09	18.10
1	3.55	6.10	8.26	10.14	11.78	13.26	14.59	15.80	16.91	17.94	18.89
2	5.70	7.92	9.83	11.51	13.01	14.37	15.60	16.73	17.77	18.74	19.64
3	7.56	9.51	11.23	12.72	14.14	15.40	16.55	17.61	18.59	19.50	20.35
4	9.20	10.95	12.51	13.91	15.19	16.36	17.44	18.44	19.36	20.23	21.03
5	10.66	12.25	13.68	14.98	16.18	17.27	18.29	19.23	20.10	20.92	21.99
6	11.99	13.45	14.77	15.98	17.10	18.13	19.09	19.98	20.81	21.59	22.32
7	13.21	14.56	15.79	16.93	17.98	18.95	19.85	20.70	21.49	22.23	22.92
8	14.34	15.60	16.75	17.82	18.81	19.73	20.59	21.38	22.14	22.84	23.50
9	15.40	16.57	17.66	18.66	19.60	20.47	21.28	22.04	22.76	23.43	24.07
10	16.39	17.50	18.52	19.47	20.35	21.18	21.95	22.68	23.36	24.00	24.61

2) The values of the completely unsuccessful batched search under the replacement model are greater than the completely successful batched search when $n \geq k^2$:

$$COST_{uns,rep} \geq COST_{suc} \text{ when } n \geq k^2.$$

3) There are breakpoints beyond which the values of the completely unsuccessful batched search under the replacement model are smaller or greater than the values of the completely successful batched search under the nonreplacement model. These breakpoints are specified by the following third order equation. That is:

$$COST_{uns,nrep} \geq COST_{uns,rep} \Leftrightarrow$$
$$k^3 - nk^2 - n^2k + (n^2 + n) = 0.$$

It is worth noting that the completely successful batched search under the nonreplacement model is always more expensive than the successful batched search under the replacement model [4].

Table III has been produced by using (4) for various numbers of existing (k_1) and nonexisting (k_2) records. The file considered contains 100 records.

B. Tree-Structured Files

Expressions (6) and (7) are exact while (8) and (9) are approximate. Tables IV and V have been produced by using (6)–(9).

Tables IV–VI concern a ternary tree with four levels. Table IV [V] is produced by using the exact formula (7) [approximate formula (9)]. Table VI shows the relative error of the approximate formula. For the parameter range used, the deviation is approximately less than 5 percent.

By following [6], this study can be extended in two possible directions. First, other approximate expressions can be derived in place of formulas (8) and (9), by taking only the first two or three factors of the summations. In this way the deviation should increase. Second, the limitation that the block capacity of the sequential file is one record, also accepted in [3], [7], can be removed. Therefore, a

TABLE V

EXPECTED VALUES OF BLOCK ACCESSES IN A COMPLETE TERNARY TREE-STRUCTURED FILE WITH 4 LEVELS FOR VARIOUS VALUES OF k_1 AND k_2 BY USING THE APPROXIMATE FORMULA (10). THE FILE CONTAINS $n = 80$ RECORDS.

		k_2										
		0	1	2	3	4	5	6	7	8	9	10
k_1	0	0.00	3.80	6.24	8.40	10.34	12.09	13.69	15.16	16.53	17.94	18.98
	1	3.37	5.81	7.97	9.91	11.67	13.28	14.76	16.13	17.40	18.60	19.72
	2	5.43	7.59	9.53	11.29	12.90	14.38	15.75	17.04	18.24	19.36	20.43
	3	7.24	9.18	10.94	12.54	14.03	15.40	16.69	17.89	19.02	20.09	21.10
	4	8.87	10.62	12.22	13.70	15.08	16.37	17.57	18.70	19.77	20.79	21.74
	5	10.34	11.94	13.41	14.97	16.07	17.27	18.41	19.48	20.49	21.45	22.36
	6	11.68	13.15	14.52	15.79	17.00	18.13	19.20	20.21	21.17	22.09	22.96
	7	12.92	14.27	15.55	16.74	17.87	18.94	19.95	20.92	21.83	22.70	23.53
	8	14.06	15.33	16.52	17.64	18.71	19.72	20.68	21.59	22.46	23.29	24.08
	9	15.13	16.32	17.44	18.50	19.50	20.46	21.37	22.24	23.07	23.86	24.62
	10	16.14	17.25	18.31	19.31	20.26	21.17	22.03	22.86	23.65	24.41	25.13

TABLE VI

RELATIVE ERROR (PERCENT) IN EXPECTED VALUES OF BLOCK ACCESSES IN A COMPLETE TERNARY TREE-STRUCTURED FILE WITH 4 LEVELS FOR VARIOUS VALUES OF k_1 AND k_2 BY USING THE EXACT (7) AND THE APPROXIMATE FORMULA (10). THE FILE CONTAINS $n = 80$ RECORDS.

		k_2										
		0	1	2	3	4	5	6	7	8	9	10
k_1	0	0.0	5.1	3.9	2.4	1.0	-0.3	-1.4	-2.4	-3.3	-4.1	-4.9
	1	5.1	4.8	3.5	2.2	1.0	-0.2	-1.2	-2.1	-2.9	-3.7	-4.4
	2	4.8	4.1	3.0	1.9	0.9	-0.1	-1.0	-1.8	-2.6	-3.3	-4.0
	3	4.1	3.5	2.6	1.7	0.8	-0.0	-0.9	-1.6	-2.3	-3.0	-3.7
	4	3.5	3.0	2.3	1.5	0.7	-0.0	-0.7	-1.4	-2.1	-2.8	-3.4
	5	3.0	2.6	2.0	1.3	0.7	0.0	-0.7	-1.3	-1.9	-2.5	-3.1
	6	2.5	2.2	1.7	1.2	0.6	0.0	-0.6	-1.2	-1.7	-2.3	-2.9
	7	2.2	1.9	1.5	1.1	0.6	0.0	-0.5	-1.1	-1.6	-2.1	-2.7
	8	1.9	1.7	1.4	1.0	0.5	0.0	-0.5	-1.0	-1.5	-2.0	-2.5
	9	1.7	1.5	1.3	0.9	0.5	0.1	-0.4	-0.9	-1.3	-1.8	-2.3
	10	1.5	1.4	1.1	0.8	0.5	0.1	-0.4	-0.8	-1.2	-1.7	-2.1

possible extension would assume block capacity greater than one and produce new formulas.

ACKNOWLEDGMENT

The authors would like to thank M. Devetsikiotis and G. Papaioannou for their help in proofreading this paper and the preparation of the tables. Also, constructive comments from the referees led to improvement of the presentation.

REFERENCES

[1] D. E. Knuth, *The Art of Computer Programming, vol. 3, Sorting and Searching*. Reading, MA: Addison-Wesley, 1973.
[2] T. J. Teorey and J. P. Fry, *Design of Database Structures*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[3] B. Shneiderman and V. Goodman, "Batched searching of sequential and tree structured files," *ACM Trans. Database Syst.*, vol. 1, pp. 268-275, 1976.
[4] S. Christodoulakis, "Estimating block transfers and join sizes," in *Proc. SIGMOD-83 Conf.*, 1983, pp. 40-54.
[5] A. F. Cardenas, "Analysis and performance of inverted database structures," *Commun. ACM*, vol. 18, pp. 255-263, 1975.
[6] P. Palvia and S. T. March, "Approximating block accesses in database organizations," Inform. *Processing Lett.*, vol. 19, pp. 75-79, 1984.
[7] D. S. Batory and C. C. Gotlieb, "A unifying model of physical databases," *ACM Trans. Database Syst.*, vol. 10, pp. 97-106, 1982.
[8] P. Palvia, "Expressions for batched searching of sequential and hierarchical files," *ACM Trans. Database Syst.*, vol. 10, pp. 97-106, 1985.
[9] S. B. Yao, "An attribute based model for database access cost analysis," *ACM Trans. Database Syst.*, vol. 2, pp. 45-67, 1977.
[10] T. Y. Cheung, "Estimating block accesses and number of records in file management," *Commun. ACM*, vol. 25, pp. 484-487, 1982.
[11] W. S. Luk, "On estimating block accesses in database operations," *Commun. ACM*, vol. 26, pp. 945-947, 1983.
[12] K. Y. Whang, G. Wiederhold, and D. Sagalowicz, "Estimating block accesses in database operations: A closed non-iterative formula," *Commun. ACM*, vol. 26, pp. 940-944, 1983.
[13] S. B. Yao, "Approximating block accesses in database organizations," *Commun. ACM*, vol. 20, pp. 260-261, 1977.
[14] M. Piwowarski, "Comments on batched searching of sequential and tree-structured files," *ACM Trans. Database Syst.*, vol. 10, pp. 285-287, 1985.
[15] Y. Manolopoulos, J. G. Kollias, and M. Hatzopoulos, "Binary vs. sequential batched search," *Comput. J.*, vol. 29, pp. 368-372, 1986.
[16] Y. Manolopoulos, L. Petrou, and D. Kleftouris, "Searching for composite queries in a main memory database," *Angewandte Informatik*, vol. 84, pp. 141-148, 1987.
[17] Y. Manolopoulos and J. G. Kollias, "Estimating disk head movement in batched search," *BIT*, vol. 28, pp. 27-36, 1988.
[18] ——, "Performance of a two-headed disk system when serving database queries under the SCAN policy," *ACM Trans. Database Syst.*, 1989.
[19] Y. Manolopoulos, "Probability distributions for seek time evaluation," *Inform. Sci.*, submitted for publication.

Yannis Manolopoulos was born in Thessaloniki, Greece. He received the B.S. and Ph.D. degrees in electrical engineering from the Aristotelian University of Thessaloniki, Greece, in 1981 and 1986, respectively. His Ph.D. degree is in the area of computer science. During the academic year 1984-1985 he was a research visitor at CSRI, University of Toronto, Toronto, Ont., Canada.

After serving his obligatory military service, he joined Aristotelian University, where he is currently an Assistant Professor. His research interests include physical database design and performance evaluation of access methods.

Dr. Manolopoulos is a member of the Association for Computing Machinery and the IEEE Computer Society.

J. (Yannis) G. Kollias was born in Patras, Greece. He received the B.S. degree in mathematics from the University of Athens, Athens, Greece, in 1968, the M.Sc. degree in computer science from the University of Newcastle-upon-Tyne, England, in 1969, and the Ph.D. degree in computer science from the University of East Anglia, Norwich, England, in 1976.

Until February 1983, he was DP manager of a bank in Greece. During the Spring term of 1983 he was a Visiting Associate Professor at the Michigan Technological University, Houghton. Since 1983 he has been a Professor at the National Technical University of Athens. His research interests include physical database design, distributed databases, and spatial processing.

Dr. Kollias is a member of the Association for Computing Machinery and the IEEE Computer Society, and a Fellow of the British Computer Society (FBCS).