Optimizing Index Allocation for Sequential Data Broadcasting in Wireless Mobile Computing

Ming-Syan Chen, *Senior Member*, *IEEE*, Kun-Lung Wu, *Member*, *IEEE Computer Society*, and Philip S. Yu, *Fellow*, *IEEE*

Abstract—Energy saving is one of the most important issues in wireless mobile computing. Among others, one viable approach to achieving energy saving is to use an indexed data organization to broadcast data over wireless channels to mobile units. Using indexed broadcasting, mobile units can be guided to the data of interest efficiently and only need to be actively listening to the broadcasting channel when the relevant information is present. In this paper, we explore the issue of indexing data with skewed access for sequential broadcasting in wireless mobile computing. We first propose methods to build index trees based on access frequencies of data records. To minimize the average cost of index probes, we consider two cases: one for fixed index fanouts and the other for variant index fanouts, and devise algorithms to construct index trees for both cases. We show that the cost of index probes can be minimized not only by employing an imbalanced index tree that is designed in accordance with data access skew, but also by exploiting variant fanouts for index nodes. Note that, even for the same index tree, different broadcasting orders of data records will lead to different average data access times. To address this issue, we develop an algorithm to determine the optimal order for sequential data broadcasting to minimize the average data access time. Performance evaluation on the algorithms proposed is conducted. Examples and remarks are given to illustrate our results.

Index Terms—Wireless mobile computing, energy saving, indexing, sequential broadcasting.

1 Introduction

Recently, there has been an increasing interest to equip the data networks with mobility [4], [10], [28]. Mobile computing, referring to the activity of using personal digital assistances such as palmtops and notebook computers to access a large number of databases via wireless networks, has been identified as an area with an emerging market [12], [18], [20], [27]. A significant amount of research effort has been elaborated upon exploring several aspects of mobile computing, including energy saving [17], [21], [25], [28], [33], cache management [3], [6], query processing [19], [29], and data allocation and communication [14], [18], [26], [31]. Such applications as using palmtops to access airline schedules, stock activities, traffic conditions, and weather information on the road are expected to become increasingly popular. It is noted, however, that several mobile computers, such as desktops and palmtops, use batteries of limited lifetime for their operations and are not directly connected to any power source. As a result, energy saving is a very important issue to resolve before we can anticipate an even wider acceptability for mobile computers [20], [24], [28].

In general, a mobile server is expected to concurrently serve hundreds or even thousands of clients. Since the cost of broadcasting is independent of the number of users, periodic broadcasting is an important method for information dissemination in a wireless communication environment [21], [30]. For energy saving, it is important for a mobile client to be able to operate in two different modes: *doze mode*, where it is still connected to the network but it is not active, and *active mode*, where it performs usual activities [30]. Explicitly, when a palmtop is actively listening to a channel, its CPU must be in an *active mode* to examine data packets so as to determine if they match the predefined patterns. Such CPU operations for data examination consume a significant amount of battery power. Therefore, to achieve energy saving it is highly desirable to let the palmtops stay in the *doze mode* most of the time and only come to the active mode when it is necessary. As a consequence, it is advantageous to use indexed data organization to broadcast data over wireless channels so that those mobile units can be guided to the data of interest efficiently and only need to be actively listening to the broadcasting channel when the relevant information (indexes or data) is present. The structure of an index tree determines the index probing scenario to switch between the doze and the active modes for data access under such an indexed broadcasting. More justifications for indexed broadcasting and some pioneering results on this aspect can be found in [1], [2], [9], [15], [16], [21], [23].

A conventional index tree is given in Fig. 1a, and its corresponding broadcasting sequence is given in Fig. 1b. Suppose that record R5 is the record to be accessed. Then, after being routed to the root index I, the request will probe I and a_2 and then reach R5. Clearly, using this index tree, a request to any record will take two index probes. Note that the amount of time a mobile unit has to stay in the active mode is proportional to the number of index probes it has to make.

- *M.-S. Chen is with the Electrical Engineering Department, National Taiwan University, Taipei, Taiwan, Republic of China. E-mail: mschen@cc.ee.ntu.edu.tw.*
- *P.S. Yu and K.-L. Wu are with IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown, NY 10598. E-mail: {psyu, klwu}@us.ibm.com.*

Manuscript received 6 Mar. 2000; revised 29 Jan. 2001; accepted 16 Oct. 2001.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 111632.

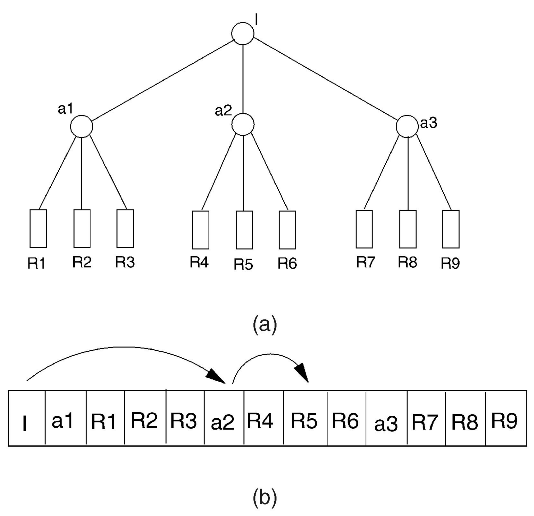

Fig. 1. Indexed broadcasting: (a) an example index tree and (b) index probing scenario to data record R_5.

Hence, reducing the average number of index probes will result in a lower power consumption. It is noted that, in most databases, the access frequencies of different data records are usually different from one another [7], [8]. This phenomenon is termed data access skew. In view of the existence of data skew, an index tree can in fact be judiciously built in accordance with data access frequencies in such a way that the average cost of index probes for data access is minimized. Notice, however, that most of the prior studies do not consider exploiting the feature of data access skew to minimize the number of index probes when an index tree is constructed. Also, there is no attempt reported to utilize variant index fanouts to minimize the average cost of index probes. Without dealing with these two features, mostly employed in prior work is a symmetric balanced index tree with essentially the same fanouts for all index nodes. The absence of research effort in this issue can be explained by the reason that, in the conventional file system, the cost of executing an index probe for data access is relatively small as compared to other CPU and I/O operations, thus, rendering little overall performance improvement by minimizing the number of index probes. Therefore, a sophisticated method to reduce the index probe cost, such as employing an imbalanced index tree or allowing variant index fanouts, is not deemed important from a performance perspective. Furthermore, the normal update activities in a file system will make the maintenance/reconstruction of index trees with variant fanouts very expensive. However, such a cost model does not hold in the context of wireless mobile computing. In wireless mobile computing, because listening to broadcast data is one of the major sources for power consumption, it is very important to minimize the cost of index probes to access the broadcast data, which are mostly for read-only applications. The importance and feasibility of employing sophisticated index trees to minimize the cost of index probes is, thus, justified in wireless mobile computing.

Consequently, we explore in this paper, the issue of indexing data with skewed access for sequential broadcasting in wireless mobile computing. Specifically, we first propose methods to build index trees based on access frequencies of data records. To minimize the average cost of index probes, we consider two cases: one for fixed index fanouts and the other for variant index fanouts. For the case of fixed index fanouts, in light of the Huffman code [22], we derive an algorithm, CF (standing for constant fanout), for the optimal index tree construction that minimizes the average number of index probes. With CF derived, we then consider a more sophisticated model in wireless mobile computing that allows variant index fanouts in an index tree. For the case of variant index fanouts, due to the NP-Completeness of the corresponding optimization problem [11], we devise an efficient greedy algorithm, VF (standing for variant fanout), for suboptimal solutions. We show that the average cost of index probes can be significantly reduced not only by employing an imbalanced index tree that is designed in accordance with data access skew, but also by exploiting variant fanouts for index nodes. Explicitly, by employing the concept of variant index fanouts, one can obtain an index tree that requires, on the average, a smaller index probe cost than optimal fixed fanout index trees. This feature, in our opinion, is particularly important for indexed broadcasting in a wireless communication environment. In order to evaluate the performance of algorithms proposed, we implement a simulation model for sequential broadcasting. Specifically, we first examine the benefit of utilizing imbalanced index trees, and then comparatively evaluate the performance of algorithms CF and VF. It is shown by the experimental results that the use of imbalanced index trees will lead to significant performance improvement over the use of the balanced index trees, and such an advantage becomes even more prominent as the skewness of data access increases.

Note that the structure of an index tree determines the average index probe cost. However, even for the same index tree, different broadcasting orders of data records will lead to different average data access times. The problem of determining the optimal order for indexed broadcasting is complicated since not only has the existence of index nodes to be considered but also the underlining indexing hierarchy has to be followed. We shall first derive some properties for index trees and then, in light of these properties, develop an algorithm ORD to determine, for a given index tree, the optimal order for sequential data broadcasting. Performance evaluation on algorithm ORD is also conducted. Examples and remarks are given to illustrate our results. It is noted that the three algorithms proposed, i.e., CF, VF, and ORD, are shown to have polynomial time complexity.

This paper is organized as follows: Preliminaries are given in Section 2. Algorithms for building index trees for both the cases of fixed fanouts and variant fanouts are derived in Section 3. Section 4 determines the optimal order of sequential broadcasting for a given index tree. This paper concludes with Section 5.

2 PRELIMINARIES

In this paper, a mobile client is assumed to use selective tuning to listen to indexed sequential data broadcasting.

TABLE 1
The Access Probability for Each Data Record and the Number of Index Probes Required to Reach Each Record by $T^B_{d=3}$ and $T^I_{d=3}$

Data record	R1	R2	R3	R4	R5	R6	R7	R8	R9
$Pr(R_i)$	0.4	0.4	0.05	0.05	0.02	0.02	0.02	0.02	0.02
$I_{pb}(R_i)$ in $T^B_{d=3}$	2	2	2	2	2	2	2	2	2
$I_{pb}(R_i)$ in $T^I_{d=3}$	1	1	2	3	3	3	3	3	3

Note that, if the data is broadcasted without any index, the client will have to listen to the channel, on the average, half of the total broadcasting time for the whole file. As mentioned earlier, using proper indexing in sequential broadcasting, selective tuning allows mobile clients to stay active only when the data of interest is present, thereby saving a lot of battery resource. As in [21], [30], we use the following two parameters, *tuning time* and *access time*, to assess the performance of a data broadcasting scheme. Tuning time means the amount of time spent by a client to listen to the channel. Listening to a channel requires the client to be in the active mode [21]. Hence, tuning time determines the battery consumption. Access time means the time elapsed from the time a client wants an identified record to the time that record is downloaded by the client. Access time further consists of two components: *probe wait* and *bcast wait*. Probe wait is the time from the point a client tunes in to the point when the first index is reached, and bcast wait is time duration from the point the first index is reached to the point the required record is obtained. Assuming that probe wait is half the length between two consecutive starting indexes, our interest in this study starts from the point that the first index is captured for ease of discussion.

Explicitly, we shall explore in Section 3, the approaches of using imbalanced index trees as well as employing variant index fanouts to minimize the average index probe cost (i.e., tuning time), thus, minimizing the battery consumption. Moreover, we shall investigate in Section 4, the optimal order of data records in sequential broadcasting to minimize the average access time.[1] Note that there are some methods, such as index replications, available to reduce the probe wait in access time, whose discussion is beyond the scope of this paper.[2] Interested readers are referred to [5], [21].

Denote the number of data records as n, and a data record as $R_i, 1 \leq i \leq n$. Let $Pr(R_i)$ be the access probability of R_i, i.e., $\sum_{1 \leq i \leq n} Pr(R_i) = 1$, $I_{pb}(R_i)$ be the number of index probes to reach R_i in an index tree. Also, a_i is used to represent an index node in an index tree and $d(a_i)$ represents the fanout of a_i. $Path(R_i)$ is the set of index nodes from the root to data record R_i. For example, we have $I_{pb}(R_3) = 2$, $d(a_1) = 3$, and $Path(R_3) = \{I, a_1\}$ in Fig. 1a.

1. As mentioned above, access time consists of probe wait and bcast wait. Since our interest starts from the point that the first index is captured. Our effort on minimizing access time focuses on minimizing bcast wait.

2. Using the terminology in [21], the index distribution model employed in this paper is "nonreplicated distribution."

3 Index Allocation for Skewed Data Access

We first propose algorithm CF to build a fixed fanout index tree based on access frequencies of data records. Then, we devise algorithm VF to construct an index tree with variant fanouts. As will be shown by experimental results, the average cost of index probes can be minimized not only by employing an imbalanced index tree that is designed in accordance with data access frequencies, but also by exploiting variant fanouts for index nodes.

3.1 Imbalanced Index Tree Construction for Fixed Fanouts

In essence, the proposed method CF will reduce the number of index probes for hot (i.e., frequently accessed) data while allowing more probes for cold (less frequently accessed) data. CF, which is devised in light of the Huffman code, is described below, where n is the number of total data records and d is the fanout of each index.

Algorithm CF: Use access frequencies to build an index tree with a fixed fanout d.

Step 1: Initially, we have a forest of n subtrees, each of which is a single node labeled with the corresponding access frequency.

Step 2: Attach the d subtrees with the smallest labels to a new node. Label the resulting subtree with the sum of all labels from its d child subtrees.

Step 3: $n = n - d + 1$. (Then, n is the number of remaining subtrees.) If $n = 1$ stop else goto Step 2.

It can be seen that, when the number of fanouts is equal to two, i.e., $d = 2$, algorithm CF becomes the conventional Huffman code algorithm [22]. To facilitate our presentation, the index tree in Fig. 1a is denoted by $T^B_{d=3}$ and the corresponding imbalanced tree built by algorithm CF is denoted by $T^I_{d=3}$. The superindexes of $T^B_{d=3}$ and $T^I_{d=3}$ stand for "balanced" and "imbalanced," respectively. For example, given the profile in Table 1 and $d = 3$, algorithm CF will build an imbalanced index tree in a bottom up manner and lead to $T^I_{d=3}$ as shown in Fig. 2a. It can be verified from Table 1 that, by using the imbalanced tree $T^I_{d=3}$, the average number of index probes for data access, i.e., $\sum_{1 \leq i \leq 9} Pr(R_i)I_{pb}(R_i)$, is reduced from two (i.e., using the index tree $T^B_{d=3}$ in Fig. 1a) to 1.5 (i.e., using the index tree $T^I_{d=3}$ in Fig. 2a), thus, reducing the average power consumption. A corresponding broadcasting sequence is given in Fig. 2b. Consequently, we have the following theorem.

Theorem 1. *Given a fixed index fanout, the average number of index probes, i.e.,* $\sum_{1 \leq i \leq n} Pr(R_i)I_{pb}(R_i)$, *is minimized by using the index tree constructed by algorithm* CF.

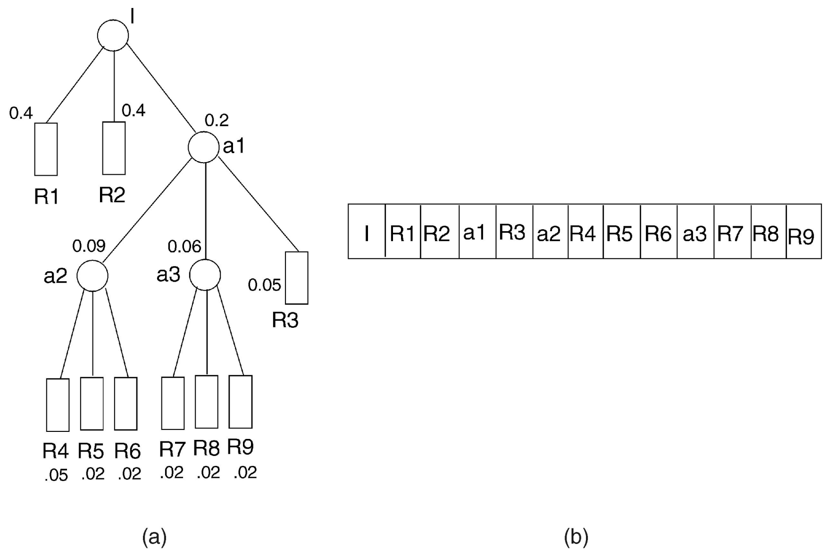

Fig. 2. Illustration of an imbalanced index tree: (a) an index tree $T^I_{d=3}$ built by CF and (b) corresponding data broadcasting sequence.

Proof. It can be seen that the formula to determine the number of index probes in an index tree with a fanout d is the same as the one of determining the weighted path length of a d-ary tree in [22]. By transforming the problem of minimizing the number of index probes to the one of determining the minimal weighted path length for a d-ary tree, algorithm CF is, in essence, a generalized version of the Huffman code algorithm which is proven in [22] to be able to lead to the minimal weighted path length for a d-ary. This theorem follows.□

Denote the cost of probing an index a_i as $f(a_i)$. Then, the average cost of locating a record by probing indexes can be expressed as:

$$\sum_{1\le i\le n} Pr(R_i) \sum_{a_j\in Path(R_i)} f(a_j),$$

where $Path(R_i)$ is the set of index nodes from the root to data record R_i. Note that, under sequential broadcasting, the cost of probing an index is proportional to the fanout of that index, i.e., $f(a_i) = g(d(a_i))$, where $g(\cdot)$ is a monotonically increasing function. Without loss of generality, we shall use $f(a_i) = g(d(a_i)) = d(a_i)$ in our discussion below for ease of presentation.[3] For example, for the index tree in Fig. 2a, $f(a_2) = d(a_2) = 3$ and the index probe cost to reach data record R_5 is

$$\textstyle\sum_{a_j\in Path(R_5)} d(a_j) = d(I) + d(a_1) + d(a_2) = 3 + 3 + 3 = 9.$$

Given this cost model, it is important to note that index trees built by algorithm CF for different fanouts will lead to different average cost of index probes. For example, index trees built by algorithm CF for $d = 2$ and $d = 4$ are given in Fig. 3 and Fig. 4, respectively. Let $\mathrm{C}(T^I_{d=j})$ be the average cost of index probes for the index tree $T^I_{d=j}$, i.e., $\mathrm{C}(T^I_{d=j})=\sum_{1\le i\le n} Pr(R_i) \sum_{a_j\in Path(R_i)} d(a_j)$ for the index tree $T^I_{d=j}$. It can be obtained from Table 2 that

$$\begin{aligned}C(T^I_{d=3}) &= 6*0.4 + 16*0.05 + 54*0.02\\ &= 4.05 < C(T^I_{d=4}) = 4.12 < C(T^I_{d=2}) = 4.28,\end{aligned}$$

showing that there is no monotonic relationship, neither increasing nor decreasing, for the values of $\mathrm{C}(T^I_{d=j})$ along the fanout j allowed. This fact, together with the hierarchical nature of index construction, implies that, by allowing variant index fanouts within an index tree, one can further minimize the average cost of index probes. This important feature is explored next.

3.2 Employing Variant Index Fanouts to Minimize Index Probes

Given the remarks on the nature of CF above, we explore in this section the approach of using variant index fanouts to

3. This assumption is made for simplicity and is not required for the algorithms developed. Relaxation on the formulation of $f(a_i)$ is given in Section 4.2.

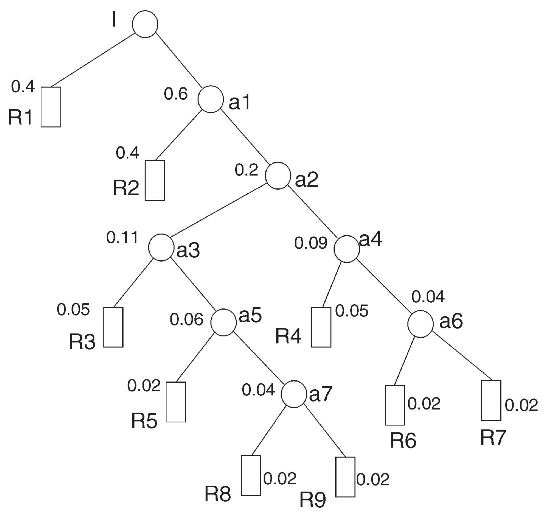

Fig. 3. An index tree $T^I_{d=2}$ built by algorithm CF.

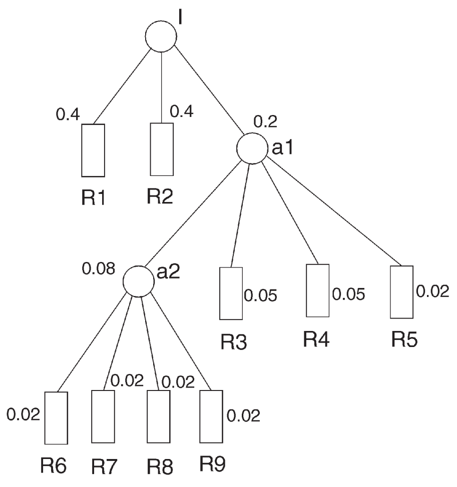

Fig. 4. An index tree $T^I_{d=4}$ built by algorithm CF.

minimize the average cost of index probes. It is noted that, unlike the conventional file system where a simple data structure is usually preferable, the approach of using variant index fanouts can be easily implemented in the indexed broadcasting environment to lead to performance improvement. As can be seen later, by employing the concept of variant index fanouts, one can obtain an index tree that requires, on the average, a smaller index probe cost than optimal fixed fanout index trees. Note that such an optimal partition problem associated with variant index fanouts is known to be NP-Complete in nature [11]. In view of this, we shall develop in the following an efficient heuristic algorithm VF to build an index tree with variant fanouts. It is observed that, on one hand, we want data records to stay as close to the root as possible, which, on its own, suggests that all data records be attached to the root node. On the other hand, however, indexes with larger fanouts are in general more costly to probe and, thus, undesirable, particularly for those data records with high access frequencies. In light of these, algorithm VF is so designed that it strikes a compromise between these conflicting factors and minimizes the average cost of index probes.

Basically, algorithm VF is greedy in nature and builds the index tree in a top down manner. VF starts with attaching all data records to the root node. Then, after some evaluation, VF groups nodes with small access frequencies and moves them to one level lower so as to minimize the average index probe cost. Fig. 5 shows the scenario of grouping a set of nodes and moving them to a lower level. It can be seen from Fig. 5 that, while the cost of index probes is increased for those nodes moved down to the next level (i.e., the index probe cost for each node among h_{i+1}, $h_{i+2}, \ldots$, and h_m is increased from m to $(i+1)+(m-i)= m+1$), the index probe cost for the other nodes will be greatly reduced (i.e., the index probe cost for each node among h_1, $h_2, \ldots$, and h_i is reduced from m to $i+1$), thereby, resulting in an overall reduction on the average index probe cost. Formally, the criterion of determining if a group of nodes should be moved to the next level can be formulated by the lemma below.

Lemma 1. *Suppose that node r has m child nodes, h_1, h_2, $\ldots$, and h_m, which are sorted according to descending order of $Pr(h_j)$, $1 \le j \le m$, i.e., $Pr(h_j) \ge Pr(h_k)$ if and only if $j \le k$. Then, the average cost of index probes can be reduced by grouping nodes h_{i+1}, h_{i+2}, $\ldots$, and h_m and attaching them under a new child node if and only if*

$$(m-i-1)\sum_{1\le j\le i} Pr(h_j) > \sum_{i+1\le j\le m} Pr(h_j).$$

Proof. Consider the average costs of index probes *before* and *after* the referenced operation (i.e., the one grouping nodes h_{i+1}, $h_{i+2}, \ldots$, and h_m and attaching them under a new child node). Use Fig. 5 for illustration and let PT(h_j) be the set of indexes from the root to h_j excluding the parent node of h_j, i.e., PT$(h_j) = Path(h_j) - \{r\}$. The average cost before this operation is

$$\begin{aligned} C_{BE} &= \sum_{1\le j\le m} Pr(h_j)(\sum_{a_p\in PT(h_j)} d(a_p) + m) \\ &= \sum_{1\le j\le i} Pr(h_j)(\sum_{a_p\in PT(h_j)} d(a_p) + m) \\ &\quad + \sum_{1+i\le j\le m} Pr(h_j)(\sum_{a_p\in PT(h_j)} d(a_p) + m), \end{aligned}$$

where m is the degree of the parent node of h_j, i.e., $d(r) = m$. In contrast, the average cost after that operation is

$$\begin{aligned} C_{AF} = &\sum_{1\le j\le i} Pr(h_j)(\sum_{a_p\in PT(h_j)} d(a_p) + (i+1)) \\ &+ \sum_{1+i\le j\le m} Pr(h_j)(\sum_{a_p\in PT(h_j)} d(a_p) + (i+1+m-i)). \end{aligned}$$

We hence, have $C_{AF} - C_{BE}$ = $(i+1-m)\sum_{1\le j\le i} Pr(h_j) + \sum_{i+1\le j\le m} Pr(h_j)$, thus proving this lemma. □

In light of Lemma 1, we devise algorithm VF below, which contains a recursive procedure *Partition* to identify the group of nodes to be moved downward in each execution level.

TABLE 2
The Index Probe Cost Required to Reach Each Record by Different Index Trees Built by CF

Data record	R1	R2	R3	R4	R5	R6	R7	R8	R9
$Pr(R_i)$	0.4	0.4	0.05	0.05	0.02	0.02	0.02	0.02	0.02
$\sum_{a_j\in Path(R_i)} d(a_j)$ in $T^I_{d=2}$	2	4	8	8	10	10	10	12	12
$\sum_{a_j\in Path(R_i)} d(a_j)$ in $T^I_{d=3}$	3	3	6	9	9	9	9	9	9
$\sum_{a_j\in Path(R_i)} d(a_j)$ in $T^I_{d=4}$	3	3	7	7	7	11	11	11	11

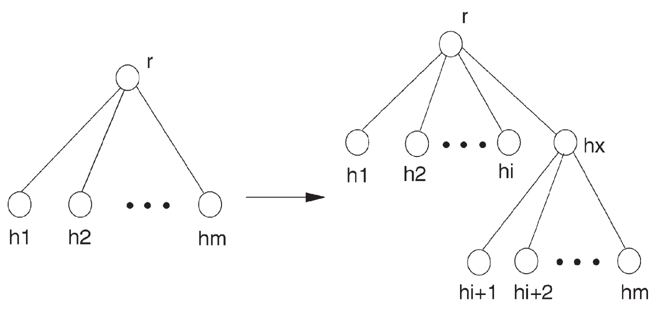

Fig. 5. Group a set of nodes and move them to a lower level to reduce the average index probe cost.

Algorithm VF:

Step 1: Assume that R_1, R_2, ..., and R_n have been sorted according to descending order of $Pr(R_j)$, $1 \leq j \leq n$, i.e., $Pr(R_j) \geq Pr(R_k)$ iff. $j \leq k$.

Step 2: *Partition* $(R_1, R_2, \ldots, R_n)$.

Step 3: Report the resulting index tree.

Procedure *Partition* first starts with a configuration where all nodes are attached to the root and uses Lemma 1 to determine if the average index probe cost can be reduced by moving a set of nodes to the next level. If there are many candidate sets of nodes whose movements are beneficial, the one with the maximal reduction on the index probe cost is chosen (i.e., operation (1) in *Partition*). When such a set of nodes is identified and moved to the next level, those nodes will be evaluated by themselves to see if any further downward movement for some of them is necessary (i.e., operation (4) in *Partition*). In addition, in the original level, by replacing those nodes moved downward with a new index node (e.g., h_x in Fig. 5) and assigning that node with an access frequency equal to the aggregate frequency of its child nodes, procedure *Partition* is called for again to see if any further downward movement for some nodes in the new list is necessary (i.e., operation (6) in *Partition*). Procedure *Partition* partitions the nodes recursively with the objective of minimizing the average index probe cost. The index tree is then constructed in a top down manner.

```
Procedure Partition $(h_1, h_2, \ldots, h_m)$:
1. Let $y(i) = (m-i-1)\sum_{1\leq j\leq i} Pr(h_j) - \sum_{i+1\leq j\leq m} Pr(h_j)$.
   Determine $i^*$ such that $y(i^*) = \max_{\forall i\in\{1,m-2\}} \{y(i)\}$.
2. If $y(i^*) \leq 0$, then return.
3. Attach nodes $h_{i^*+1}, h_{i^*+2}, \ldots, h_m$ under a new index node
   $hx$ in the index tree.
4. Partition $(h_{i^*+1}, h_{i^*+2}, \ldots, h_m)$.
5. Insert $hx$ into the ordered list $(h_1, h_2, \ldots, h_{i^*})$ and relabel
   them as $(h_1, h_2, \ldots, h_{i^*+1})$ according to descending order
   of $Pr(h_j)$, $1 \leq j \leq i^*+1$.
6. Partition $(h_1, h_2, \ldots, h_{i^*+1})$.
7. Return.
```

For example, consider the profile in Table 3. The initial index tree configuration in shown in Fig. 6a, where all data records are attached to the root. Procedure *Partition* then determines the optimal group of nodes to be moved to the next level. From the calculations shown in Table 4, we obtain $i^* = 4$ and, therefore, group nodes R_5, $R_6, \ldots$, and R_{11} together, and move them to the next level, resulting in the configuration shown in Fig. 6b. Nodes R_5, $R_6, \ldots$, and R_{11} under the index node a_1 are then partitioned recursively as shown in Fig. 7. Also, in the original level, nodes R_5, $R_6, \ldots$, and R_{11} are now replaced with a_1 which is assigned with an access frequency of 0.12. We next determine if a further partition for the new list of child nodes under the root (i.e., R_1, R_2, R_3, R_4, and a_1) is necessary. Following the calculations shown in Table 5, we obtain Fig. 6c, which in turn leads to Fig. 6d. Consequently, by combining Fig. 6 and Fig. 7, we obtain the resulting index tree T_V^I in Fig. 8. For comparison purposes, we obtain from algorithm CF the optimal index trees with fixed fanouts $T_{d=2}^I$ and $T_{d=3}^I$, shown in Fig. 9, for the same data profile. It can be verified from Fig. 9 and Table 3 that

$$\mathrm{C}(T_{d=2}^I) = \sum_{1\leq i\leq 11} Pr(R_i) \sum_{a_j\in Path(R_i)} d(a_j) = 5.2$$

and $\mathrm{C}(T_{d=3}^I) = 5.625$, both larger than $\mathrm{C}(T_V^I) = 5.08$, showing that the index tree with variant fanouts obtained by VF requires a smaller average index probe cost than optimal fixed fanout index trees.

3.3 Experimental Results on Index Allocation

In order to evaluate the performance of algorithms CF and VF proposed, we have implemented a simulation model for sequential broadcasting. Specifically, we examine the benefit of utilizing imbalanced index trees first, and then comparatively evaluate the performance of algorithms CF and VF.

Table 6 summarizes the meanings for some simulation parameters. The number of data records to be broadcasted is denoted by n and the number of fanouts allowed in each index node is d. The access frequencies of data items are modelled by the Zipf distribution [13]. Let $Pr(R_{h_i}) = (\frac{n-i}{n})^\theta$, where θ is the skew factor and R_{h_i}, $1 \leq i \leq n$, are data items sorted in descending order of access frequencies. Clearly, the access frequencies become increasingly skewed as the value of θ increases. For comparison purposes, a scheme

TABLE 3
The Index Probe Cost Required to Reach Each Record by Different Index Trees

Data record	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
$Pr(R_i)$	0.28	0.2	0.2	0.2	0.04	0.04	0.02	0.005	0.005	0.005	0.005
$\sum_{a_j\in Path(R_i)} d(a_j)$ in T_V^I	2	4	8	8	10	10	10	12	12	12	12
$\sum_{a_j\in Path(R_i)} d(a_j)$ in $T_{d=2}^I$	4	4	4	6	8	10	12	16	16	16	16
$\sum_{a_j\in Path(R_i)} d(a_j)$ in $T_{d=3}^I$	3	3	6	6	9	9	12	12	15	15	15

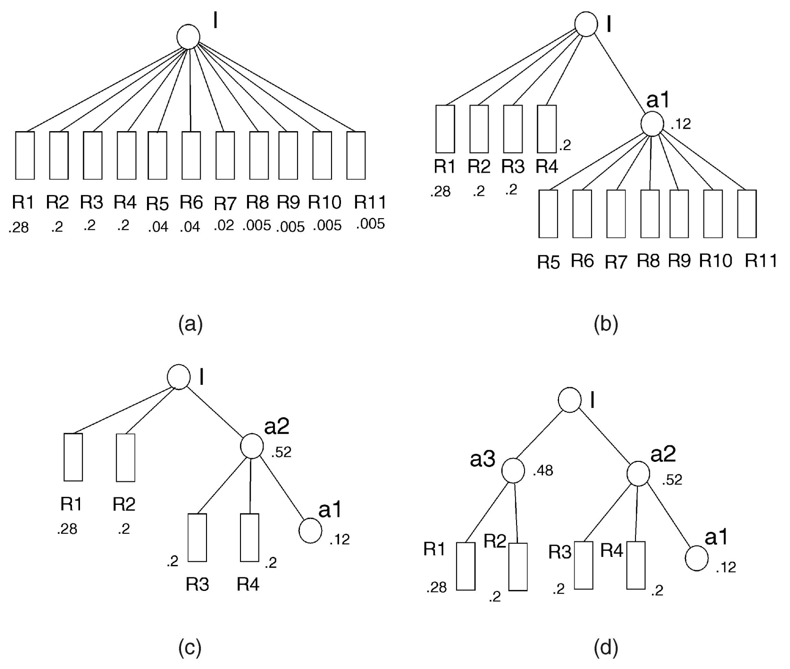

Fig. 6. Illustration of algorithm VF.

TABLE 4
Determining the Set of Nodes to be Grouped Together in Fig. 6a, where $m = 11$

i	1	2	3	4	5	6	7	8	9
$(10-i)\sum_{1<j<i} Pr(h_j)$	9*.208	8*0.48	7*0.68	6*0.88	5*0.92	4*0.96	3*0.98	2*0.985	0.99
$\sum_{i+1<j<11} Pr(h_j)$	0.72	0.52	0.32	0.12	0.08	0.04	0.02	0.015	0.01
$y(i)$	1.8	3.32	4.44	5.16	4.52	3.8	2.92	1.955	0.98

that randomly selects data items to construct a balanced index tree, referred to as BAL, is implemented.

From Fig. 10, where y-axis corresponds to the average index probe cost and x-axis corresponds to the skew factor θ, it can be seen that algorithm CF significantly outperforms BAL for various values of θ. The advantage of CF over BAL increases as the value of θ increases. This agrees with our intuition since algorithm CF is designed to take the skewness of data access into consideration. Performance of algorithms CF and VF is comparatively evaluated in Fig. 11. It is shown that, by exploring the feature of variant fanouts, algorithm VF consistently outperforms algorithm CF for various values of θ. It is seen that the advantage of VF over CF becomes even more prominent when the value of d increases.

In addition, the performance of algorithms CF and VF is comparatively evaluated with the value of n varied. Without loss of generality, the value of θ is set to two in this experiment. The corresponding results are shown in Fig. 12, where the y-axis corresponds to the average index probe cost and x-axis corresponds to the number of data items n. Again, it is shown in Fig. 12 that algorithm VF consistently outperforms algorithm CF for various values of n, showing the stability of the performance of algorithms proposed.

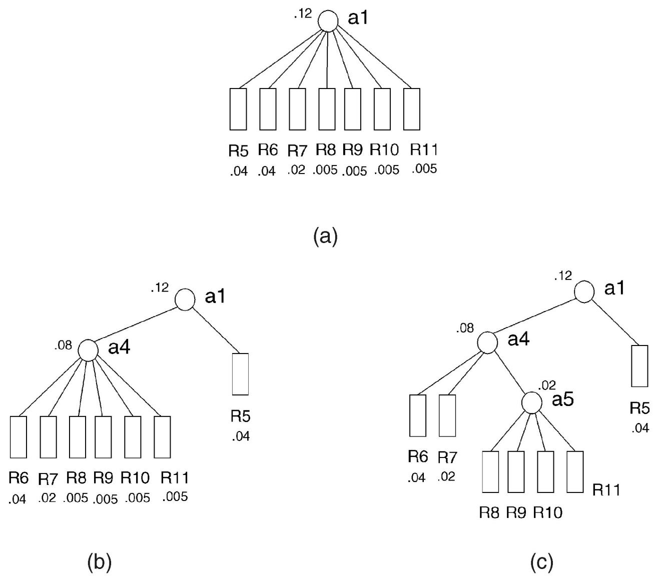

Fig. 7. Further partitions of nodes under index a_1.

TABLE 5
Determining the Set of Nodes to be Grouped Together in Fig. 6b, where $m = 5$

i	1	2	3
$(4-i)\sum_{1<j<i} Pr(h_j)$	3*0.28	2*0.48	0.68
$\sum_{i+1<j<5} Pr(h_j)$	0.72	0.52	0.32
$y(i)$	0.12	0.44	0.32

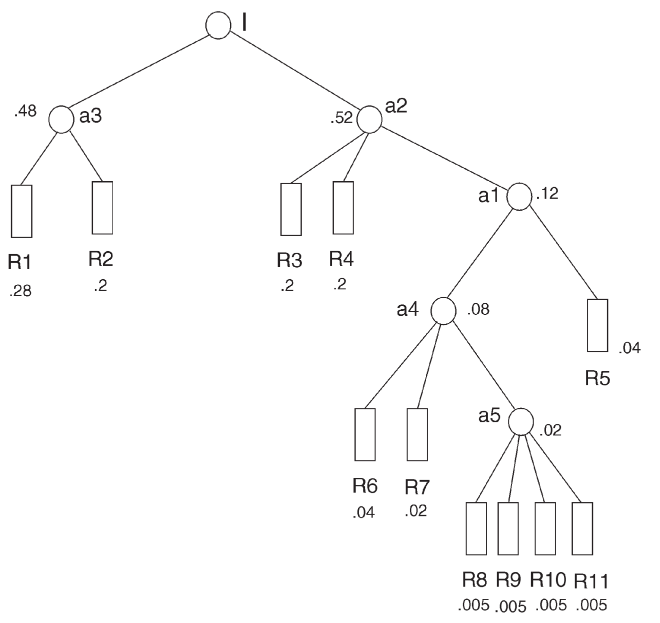

Fig. 8. The resulting index tree T^I_V (with variant fanouts) from algorithm VF.

4 Optimal Order for Sequential Data Broadcasting

As mentioned earlier, the structure of an index tree determines the average index probe cost. However, even for the same index tree, different broadcasting orders will lead to different average data access times. In Section 4.1, we derive the optimal order of sequential data broadcasting that minimizes the average data access time. Performance studies are conducted in Section 4.2. Remarks are made in Section 4.3.

4.1 Ordering Broadcasting Data to Minimize Data Access Time

Clearly, if there were no index node employed in the sequential broadcasting, we would simply broadcast data records in descending order of their access frequencies so as to minimize the average access time, assuming that mobile units start listening from the beginning of the broadcasting. Given an index tree, the optimal broadcasting sequence in this paper, refers to the one to minimize the average access time (or precisely, the one to minimize the bcast wait as explained in Section 2). This optimization however becomes very complicated in indexed broadcasting where not only does the existence of index nodes have to be considered but also the underlining indexing hierarchy has to be followed. To address this issue, we shall first derive some properties for index trees and then, in light of these properties, develop an algorithm ORD to determine for a given index tree the optimal order for sequential data broadcasting. Recall that $Path(R_i)$ is the set of index nodes from the root to data record R_i. Using this notation, we have the following property for an index tree.

Property H: *An index tree is said to possess Property H if and only if, for any two data records R_i and R_j in the index tree, $Path(R_i) \subseteq Path(R_j)$ implies $Pr(R_j) \leq Pr(R_i)$.*

Property H then leads to the following lemma.

Lemma 2. *Suppose R_i and R_j are two data records in an index tree with Property H. Then, in the optimal broadcasting sequence, data record R_i is broadcasted before R_j if $Path(R_i) \subseteq Path(R_j)$.*

Proof: We shall prove this lemma by contradiction. Suppose $Path(R_i) \subseteq Path(R_j)$ in the index tree and R_j is broadcasted before R_i in the optimal broadcasting sequence s.

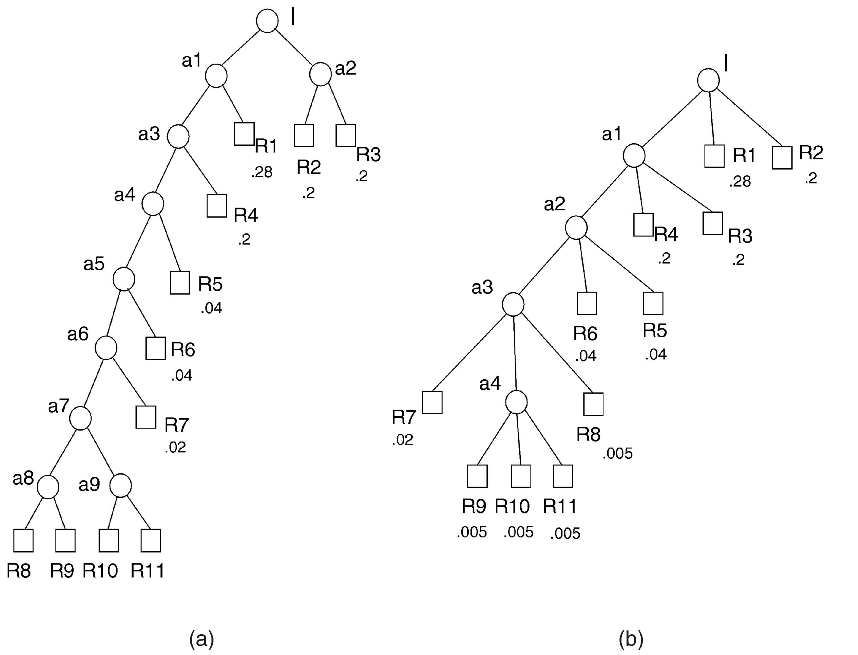

Fig. 9. (a) $T^I_{d=2}$ and (b) $T^I_{d=3}$ for the profile in Table 3.

TABLE 6
The Meanings of Some Parameters Used in the Simulation

Notation	Definition
n	Total number of data items to be broadcasted
d	Number of fanouts allowed in the index tree
θ	Zipf distribution parameter
BAL	Scheme to randomly generate a balanced index tree

Since $Path(R_i) \subseteq Path(R_j)$, implying that R_i only needs a proper subset of indexes required by R_j, we can obtain another broadcasting sequence $s*$ by interchanging R_i and R_j. From Property H, we have $Pr(R_j) \leq Pr(R_i)$, meaning that the average access time of $s*$ is less than that of s. This leads to a contradiction. This lemma follows. □

It is important to note that Property H is essential for the validity of Lemma 2. Explicitly, for an index tree without Property H, it is not always beneficial to broadcast a data record requiring fewer index nodes first. Neither is it true that a data record with higher access frequency should always precede the one with a lower access frequency. An illustrative example for these phenomena can be found in Fig. 13, where, by letting $Pr(R_1) > Pr(R_2)$, the index tree in Fig. 13a does not possess Property H. Assuming that the size of an index node is one-tenth of that of an data record, it can be verified that, when

$$\{Pr(R_1), Pr(R_2), Pr(R_3), Pr(R_4)\} = \{0.26, 0.25, 0.25, 0.24\},$$

the broadcasting sequence in Fig. 13b is favorable in terms of minimizing the average access time, where R_2 precedes R_1 despite the latter has a higher access frequency. On the other hand, when

$$\{Pr(R_1), Pr(R_2), Pr(R_3), Pr(R_4)\} = \{0.97, 0.01, 0.01, 0.01\},$$

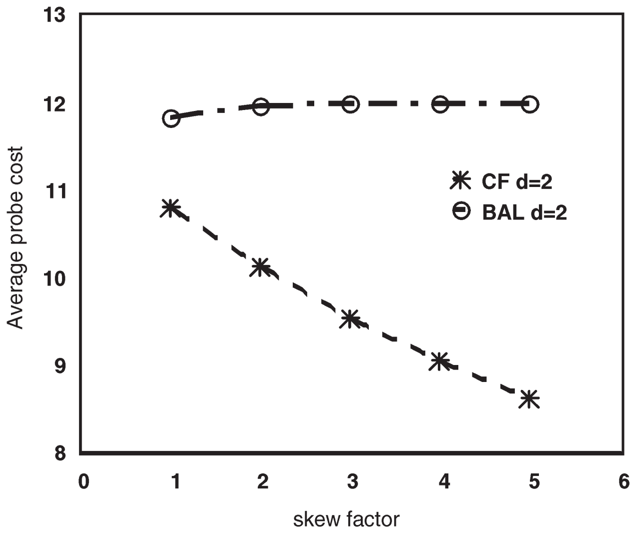

Fig. 10. Performance comparison between BAL and CF for various skew factors when $d = 2$.

the one in Fig. 13c is favorable, where R_1 precedes R_2 even the latter requires fewer index nodes. The intrinsic complexity of this problem is the very reason that we shall focus on index trees with Property H for deriving an algorithm to determine the data broadcasting order. It can be seen from the index tree construction by CF (i.e., bottom up operations) and VF (i.e., top down operations) that the index trees constructed by both algorithms possess Property H. In light of Property H and Lemma 2, we develop algorithm ORD below to determine the optimal order for sequential data broadcasting.

Algorithm ORD: To determine the optimal order for sequential data broadcasting with a given index tree.

Step 1: Denote the given index tree as T. Form the initial broadcasting sequence SEQ by data records which are sorted in descending order of their access probabilities, i.e., SEQ $= (R_1, R_2, \ldots, R_n)$, i.e., $Pr(R_j) \geq Pr(R_k)$ if and only if $j \leq k$.

Step 2: Select an index node, say a_i, from the lowest level of the index tree T (i.e., all child nodes of a_i are leaves in T). Remove the child nodes of a_i from T (i.e., a_i thus becomes a leaf node).

Step 3: Insert a_i into SEQ and place it immediately in front of its child node with the largest access frequency.

Step 4: Goto Step 2 until T becomes a single node.

For example, it can be obtained that, given the index tree in Fig. 8, algorithm ORD interleaves the sorted data records with index nodes as shown in Fig. 14a, leading to the resulting broadcasting in Fig. 14b. With data items preceding index nodes, ORD can be viewed as a pr-order search along the corresponding index tree. Note that, in ORD, every index node is always broadcasted immediately before its child nodes so that the extra access time of other nodes, which is incurred due to the presence of this index node, is minimized. Also, although Property H only characterizes the relationship among nodes within one tree branch, it can be verified that, when two branches are merged (e.g., branches under a_2 and a_3 in Fig. 8), the average access time is minimized provided that a sort-merge is performed based on the access frequencies of data records. These facts and Lemma 2 guarantee the optimality of ORD, which is formally stated below.

Theorem 2. *Given index trees built by algorithms CF and VF, the average access time is minimized by using the broadcasting order determined by algorithm ORD.*

Proof. It is noted that, in algorithm CF, the index tree is built from data records with smaller access frequencies first in a bottom up manner. Also, in algorithm VF, the index

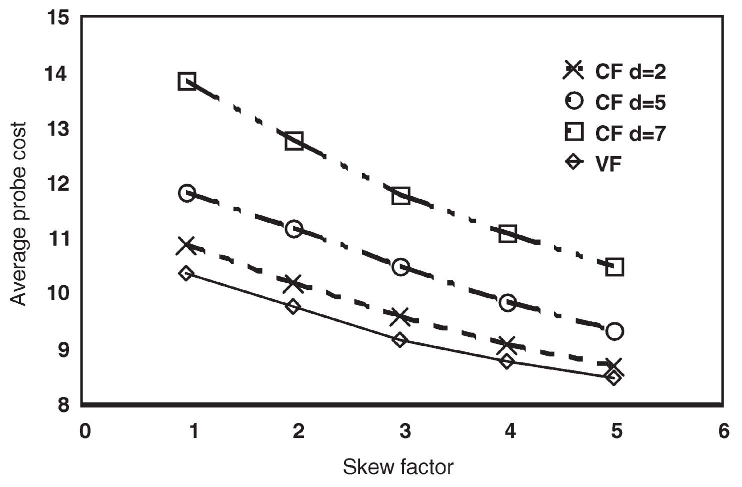

Fig. 11. Performance comparison between CF and VF for various skew factors.

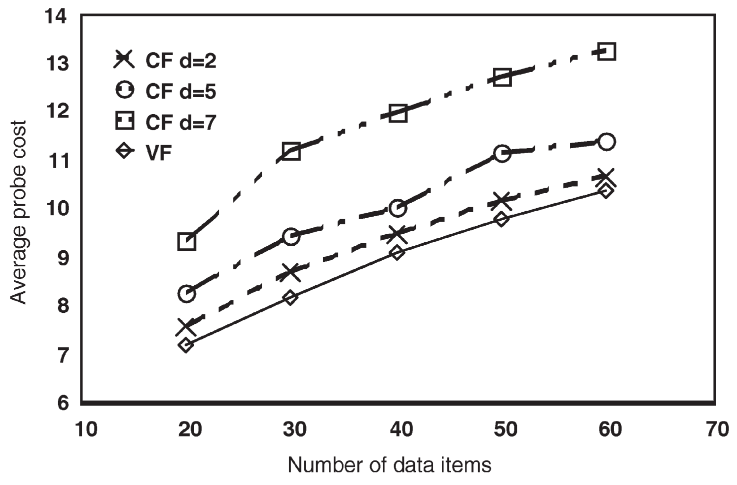

Fig. 12. Performance comparison between CF and VF for various numbers of data items.

tree is expanded by moving data records with smaller access frequencies to one level lower in a top down manner. It thus follows from both scenarios that the index trees built by algorithms CF and VF are of Property H. Then, from Lemma 2, we know that the relative order of data items in the broadcasting sequence resulted by algorithm ORD is the same as the one in the optimal broadcasting sequence. (Otherwise, an interchange of positions for a pair of out-of-order data records will render a broadcasting sequence of smaller average access time.) This theorem, hence, follows from the fact that each index node is placed immediately before its child node with the largest access frequency. □

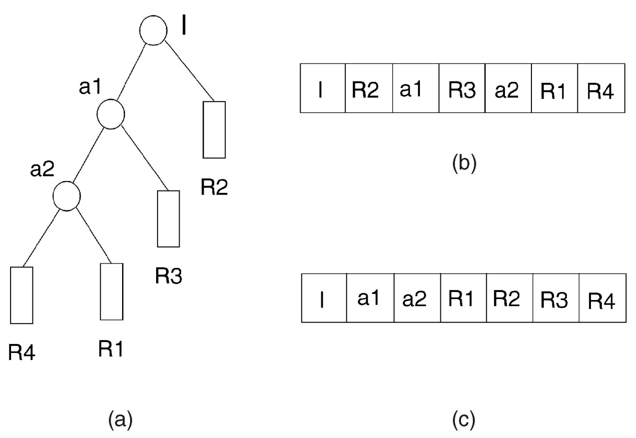

Fig. 13. An example index tree which does not possess Property H.

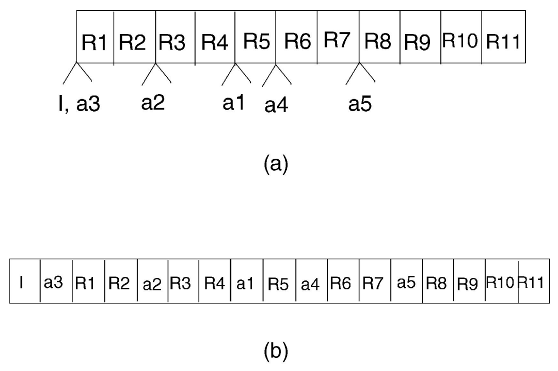
R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11
I, a3
a2
a1
a4
a5
(a)
I a3 R1 R2 a2 R3 R4 a1 R5 a4 R6 R7 a5 R8 R9 R10 R11
(b)

Fig. 14. Interleaving the broadcasting sequence with index nodes to minimize the average access time.

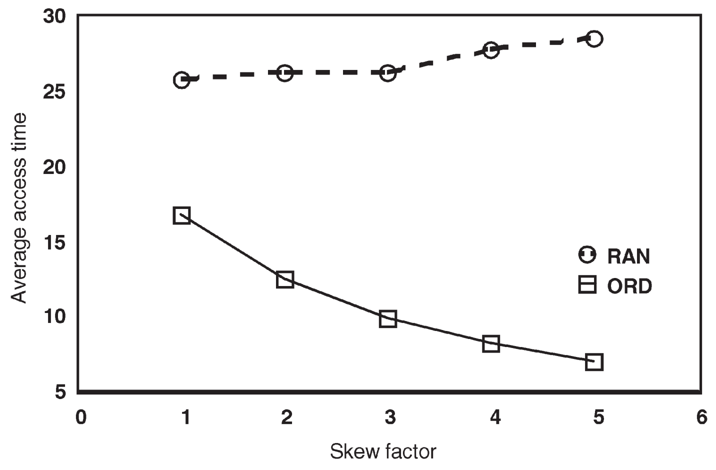

Fig. 15. Performance comparison between RAN and ORD for various skew factors.

It is noted that, in a broadcasting sequence resulting from ORD, unlike data records, index nodes may not appear in descending order of their access frequencies. An example scenario can be found in Fig. 14, where a_3 (with access frequency 0.48) precedes a_2 (with access frequency 0.52). In summary, we perform the following steps to determine the index allocation for sequential broadcasting. First, the access frequencies for data records are collected. Then, we can either use algorithm CF to build an index tree with fixed fanouts or use algorithm VF to build an index tree with variant fanouts with the objective of minimizing the average index probe cost. Finally, for an index tree built, we employ ORD to determine the optimal data broadcasting sequence to minimize the average data access time.

4.2 Experimental Results on Order of Broadcasting

In order to evaluate the performance of algorithm ORD, we conducted simulations to determine the average access time for index trees obtained by algorithm VF. Again, the number of data records to be broadcasted is n and the access frequencies of data items are modelled by the Zipf distribution. A scheme that randomly orders data items to broadcast, referred to as RAN, is implemented for comparison purposes.

Performance of algorithms ORD and RAN is comparatively evaluated in Fig. 15, where the y-axis corresponds to the average access time and the x-axis corresponds to the value of skew factor. It is shown that ORD significantly outperforms RAN due to the proper order of data records to broadcast by ORD. It is important to see that the advantage of ORD over RAN becomes even more prominent when the value of skew factor increases, showing the proper ordering is even more beneficial when the data access is more skewed. In addition, performance of algorithms ORD and RAN is evaluated for various numbers of data items. Without loss of generality, the value of θ is set to two in this experiment. The corresponding results are shown in Fig. 16, where it can be seen that algorithm ORD consistently outperforms algorithm RAN for various values of n. The advantage of ORD over RAN increases as the value of n increases.

4.3 Remarks

It can be verified that the three algorithms devised are of polynomial time complexity. Specifically, given n data records to be broadcasted, the complexity of CF is

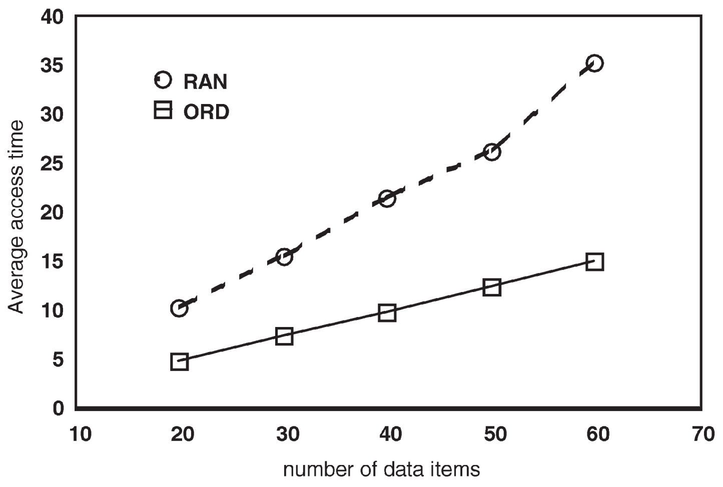

Fig. 16. Performance comparison between RAN and ORD for various numbers of data items.

O($n \log n$), that of VF is O($n^2 \log n$), and that of ORD is O($n \log n$). While the complexity of CF and ORD is dominated by sorting, VF requires O($n^2 \log n$) complexity due to its recursive operations. Note that the determination of different fanouts required in VF is performed after the sorting and incurs little computational complexity by itself. Despite its simplicity, VF usually leads to solutions with smaller costs of index probes, outperforming those optimal ones for fixed fanouts obtained by CF. In addition, as mentioned before, the assumption for $f(a_i) = d(a_i)$ we have used is made for ease of discussion, and by no means required for the applicability of VF. It can be verified that VF is valid as long as $f(a_i)$ is a monotonically increasing function with d(a_i) (i.e., an index with a larger number of fanouts requires a higher probe cost), which is reasonable in practice. By denoting $f(a_i) = g(d(a_i))$ in general, we can reformulate $y(i)$ in operation (1) of procedure *Partition* as:

$$y(i) = (g(m) - g(i+1)) \sum_{1 \le j \le i} Pr(h_j) - (g(i+1) + g(m-i) - g(m)) \sum_{i+1 \le j \le m} Pr(h_j).$$

It is worth mentioning that there are many other approaches conceivable to further reduce the data access time. In addition to replicating indexes as pointed out in [21], one may also replicate data records with high access frequencies. Note that the replication of a hot data record, while reducing the access time to that record, will lengthen the overall broadcasting cycle and, thus, increase the access time to other records. Clearly, certain criteria are needed to determine the replicated data placement and to optimize the trade-off between these conflicting factors [18], [30], [32].

5 Conclusion

In this paper, we explored the issue of indexing skewed data for sequential broadcasting in wireless mobile computing. We proposed methods to build index trees based on access frequencies of data records. Two cases, one for fixed index fanouts and the other for variant index fanouts, were considered for the minimization of the average cost of index probes. We devised algorithms CF and VF for constructing index trees for both cases. We showed, by experimental results, that the average cost of index probes can be minimized not only by employing an imbalanced index tree that is designed in accordance with data access skew, but also by exploiting variant fanouts for index nodes. Explicitly, by employing the concept of variant index fanouts, VF can lead to an index tree that requires a smaller average cost of index probes than optimal fixed fanout index trees obtained by CF. In addition, we have developed algorithm ORD to determine the optimal order for sequential data broadcasting with a given index tree. Performance evaluation on the algorithms proposed has been conducted. Complexity of the three algorithms that were proposed is analyzed. Examples and remarks were given to illustrate our results.

Acknowledgments

Ming-Syan Chen was supported in part by the Ministry of Education Project No. 89-E-FA06-2-4-7 and the National Science Council, Project No. NSC 90-2213-E-002-086, Taiwan, Republic of China. The authors would like to thank the anonymous referees for their helpful comments to improve this paper.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, “Broadcast Disks: Data Management for Asymmetric Communication Environments,” *Proc. ACM SIGMOD*, pp. 199-210, May 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik, “Dissemination-Based Data Delivery Using Broadcast Disks,” *IEEE Personal Comm.*, Dec. 1995.

[3] D. Barbara and T. Imielinski, “Sleepers and Workaholics: Caching Strategies in Mobile Environments,” *Proc. ACM SIGMOD*, pp. 1-12, May 1994.

[4] B. Bruegge and B. Bennington, “Applications of Mobile Computing and Communication,” *IEEE Personal Comm.*, pp. 64-71, Feb. 1996.

[5] J. Cai, “Information Retrieval in a Wireless Mobile Computing Environment.” PhD thesis, 1999.

[6] M.J. Carey, M.J. Franklin, M. Livny, and E. Shekita, “Data Caching Tradeoffs in Client-Server DBMS Architectures,” *Proc. ACM SIGMOD*, pp. 357-366, May 1991.

[7] M.-S. Chen, P.S. Yu, and T.-H. Yang, “On Coupling Multiple Systems with A Global Buffer,” *IEEE Trans. Knowledge and Data Eng.*, vol. 8, no. 2, pp. 339-344, Apr. 1996.

[8] A. Dan, D.M. Dias, and P.S. Yu, “The Effect of Skewed Data Access on Buffer Hits and Data Contention in a Data Sharing Environment,” *Proc. 16th Very Large Databases Conf.*, pp. 419-431, Aug. 1990.

[9] A. Datta, A. Celik, J. Kim, D. VanderMeet, and V. Kumar, “Adaptive Broadcast Protocols to Support Power Conservant Retrieval by Mobile Users,” *Proc. 13th Int'l Conf. Data Eng.*, pp. 124-133, Apr. 1997.

[10] A. Elmagarmid, J. Jain, and T. Furukawa, “Wireless Client/Server Computing for Personal Information Services amd Applications,” *Proc. ACM SIGMOD RECORD*, vol. 24, no. 4, pp. 16-21, Dec. 1995.

[11] M.R. Garey and D.S. Johnson, “Computers and Intractability: A Guide to the Theory of NP-Completeness,” 1979.

[12] D.J. Goodman, “Trends in Cellular and Cordless Communications,” *IEEE Comm. Magazine*, June 1991.

[13] J. Gray, P. Sundaresan, S. Englert, K. Baclawski, and P.J. Weinberger, “Quickly Generating Billion-Record Synthetic Databases,” *Proc. ACM SIGMOD*, pp. 243-252, Mar. 1994.

[14] P. Honeyman and L.B. Huston, “Communications and Consistency in Mobile File Systems,” *IEEE Personal Comm.*, Dec. 1995.

[15] Q. Hu, D. Lee, and W.-C. Lee, “Optimal Channel Allocation for Data Dissemination in Mobile Computing Environments,” *Proc. 18th IEEE Int'l Conf. Distributed Computing Systems*, pp. 480-487, 1998.

[16] Q. Hu, W.-C. Lee, and D. Lee, “Indexing Techniques for Wireless Data Broadcast under Data Clustering and Scheduling,” *Proc. Eighth Int'l Conf. Information and Knowledge Management*, pp. 351-358, Nov. 1999.

[17] J.-L. Huang, W.-C. Peng, and M.-S. Chen, “Binary Interpolation Search for Solution Mapping on Broadcast and On-Demand Channels in a Mobile Computing Environment,” *Proc. 10th ACM Int'l Conf. Knowledge Management*, Nov. 2001.

[18] Y. Huang, P. Sistla, and O. Wolfson, “Data Replication for Mobile Computers,” *Proc. ACM SIGMOD*, pp. 13-24, May 1994.

[19] T. Imielinski and B.R. Badrinath, “Querying in Highly Mobile Distributed Environments,” *Proc. 18th Int'l Conf. Very Large Databases*, pp. 41-52, Aug. 1992.

[20] T. Imielinski and B.R. Badrinath, “Wireless Mobile Computing: Challenges in Data Management,” *Comm. ACM*, vol. 37, no. 10, pp. 19-28, Oct. 1994.

[21] T. Imielinski, S. Viswanathan, and B.R. Badrinath, “Data on Air: Organization and Access,” *IEEE Trans. Knowledge and Data Eng.*, vol. 9, no. 3, pp. 353-372, June 1997.

[22] D. Knuth, *The Art of Computer Programming, Volume 1: Fundamental Algorithms.* Boston, Mass.: Addison Wesley, 1973.

[23] S.-C. Lo and A.L.P. Chen, “Optimal Index and Data Allocation in Multiple Broadcast Channels,” *Proc. 16th IEEE Int'l Conf. Data Eng.*, pp. 293-302, Mar. 2000.

[24] J.R. Lorch and A.J. Smith, "Software Strategies for Portable Computer Engergy Management," *IEEE Personal Comm.*, vol. 5, no. 3, June 1998.
[25] W.-C. Peng and M.-S. Chen, "Dynamic Generation of Data Broadcasting Programs for a Broadcast Disk Array in a Mobile Computing Environment," *Proc. Ninth ACM Int'l Conf. Knowledge Management*, pp. 38-45, Nov. 2000.
[26] W.-C. Peng and M.-S. Chen, "Developing Data Allocation Schemes by Incremental Mining of User Moving Patterns in a Mobile Computing System," *IEEE Trans. Knowledge and Data Eng.*, 2002.
[27] E. Pitoura and G. Samaras, *Data Management for Mobile Computing.* Kluwer Academic, 1998.
[28] S. Sheng, A. Chandrasekaran, and R.W. Broderson, "A Portable Multimedia Terminal for Personal Communications," *IEEE Comm. Magazine*, pp. 64-75, Dec. 1992.
[29] A.P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects," *Proc. 13th Int'l Conf. Data Eng.*, pp. 422-432, Apr. 1997.
[30] K.-L. Tan and B.-C. Ooi, *Data Dissemination in Wireless Computing Environments.* Kluwer Academic, 2000.
[31] O. Wolfson, S. Jajodia, and Y. Huang, "An Adaptive Data Replication Algorithm," *ACM Trans. Database Systems*, vol. 22, no. 4, pp. 255-314, June 1997.
[32] O. Wolfson and A. Milo, "The Multicast Policy and Its Relationship to Replicated Data Placement," *ACM Trans. Database Systems*, vol. 16, no. 1, pp. 181-205, Mar. 1991.
[33] K.-L. Wu, P.S. Yu, and M.-S. Chen, "Energy-Efficient Caching for Bandwidth-Limited Wireless Mobile Computing," *Proc. 12th IEEE Int'l Conf. Data Eng.*, pp. 335-343, Feb. 1996.

Ming-Syan Chen received the BS degree in electrical engineering from National Taiwan University, Taipei, Taiwan, and the MS and PhD degrees in computer, information, and control engineering from The University of Michigan, Ann Arbor, Michigan, in 1985 and 1988, respectively. Dr. Chen is currently a professor in the Electrical Engineering Department, National Taiwan University, Taipei, Taiwan. He was a research staff member at IBM Thomas J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996. His research interests include database systems, data mining, mobile computing systems, and multimedia networking, and he has published more than 140 papers in his research areas. In addition to serving as program committee members in many conferences, Dr. Chen is an associate editor of *IEEE Transactions on Knowledge and Data Engineering* on data mining and parallel database areas from 1997 to 2001, an editor of *Journal of Information Science and Engineering*, a distinguished visitor of IEEE Computer Society for Asia-Pacific from 1998 to 2000, and program chair of PAKDD-02 (Pacific Area Knowledge Discovery and Data Mining), program vice-chair of VLDB-2002 (Very Large Data Bases), general chair of Real-Time Multimedia System Workshop in 2001, program chair of IEEE ICDCS Workshop on Knowledge Discovery and Data Mining in the World Wide Web in 2000, and program co-chairs of the International Conference on Mobile Data Management in 2003, and also of International Computer Symposium on Computer Networks, Internet and Multimedia in 1998 and 2000. He was a keynote speaker on Web data mining in International Computer Congress in Hong Kong, 1999, a tutorial speaker on Web data mining in DASFAA-1999 and on parallel databases in the 11th IEEE International Conference on Data Engineering in 1995, and also a guest coeditor for the *IEEE Transactions on Knowledge and Data Engineering* special issue for data mining in December 1996. He holds, or has applied for, 18 US patents and seven Republic Of China patents in the areas of data mining, Web applications, interactive video playout, video server design, and concurrency and coherency control protocols. He received the Outstanding Innovation Award from IBM Corporation in 1994 for his contribution to parallel transaction design and implementation for a major database product, and numerous awards for his research, teaching, inventions, and patent applications. Dr. Chen is a senior member of IEEE and a member of ACM.

Kun-Lung Wu received the BS degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, and the MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign. Since 1990, he has been with the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, where he is a member of the Software Tools and Techniques Group. His current research interests include continual queries, change management, caching and other infrastructure design issues of the Web, personalization and data mining tools for the Web, Internet applications, and pervasive computing. He has published extensively in the areas of Web caching, database performance, disk subsystems, transaction and query processing, multimedia systems, mobile computing, and reliable computing. Dr. Wu has published more than 55 papers in refereed journals and conferences and over 35 research reports. He holds or has applied for 18 US patents. Dr. Wu is a member of the ACM and the IEEE Computer Society. He is currently on the editorial board of the *IEEE Transactions on Knowledge and Data Engineering*, serving as an associate editor. He was the general chair for the Third International Workshop on E-Commerce and Web-Based Information Systems (WECWIS 2001). He has served as an organizing and program committee member on various conferences. He has received various IBM awards, including Research Division Award and Invention Achievement Awards.

Philip S. Yu received the BS degree in electrical engineering from National Taiwan University, the MS and PhD degrees in electrical engineering from Stanford University, and the MBA degree from New York University. He is with the IBM Thomas J. Watson Research Center and currently manager of the Software Tools and Techniques group. His research interests include data mining, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, disk arrays, computer architecture, performance modeling, and workload analysis. Dr. Yu has published more than 320 papers in refereed journals and conferences. He holds or has applied for 244 US patents. Dr. Yu is a fellow of the ACM and a fellow of the IEEE. He is the editor-in-chief of *IEEE Transactions on Knowledge and Data Engineering*. He is also an associate editor of *ACM Transactions on the Internet Technology and that of Knowledge and Information Systems*. He is a member of the IEEE Data Engineering Steering Committee and is also on the steering committee of IEEE Conference on Data Mining. He was an editor and advisory board member of *IEEE Transactions on Knowledge and Data Engineering* and also a guest coeditor of the special issue on mining of databases. In addition to serving as program committee member on various conferences, he was the program cochair of the 11th International Conference on Data Engineering and the program chairs of the Second International Workshop on Research Issues on Data Engineering: Transaction and Query Processing, the PAKDD Workshop on Knowledge Discovery from Advanced Databases, and the Second International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems. He served as the general chair of the 14th International Conference on Data Engineering and is currently serving as the general cochair of the Second IEEE International Conference on Data Mining. He has received several IBM and external honors including a Best Paper Award, two IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, two Research Division Awards, and the 71th plateau of Invention Achievement Award. He also received an IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999. Dr. Yu is an IBM Master Inventor and was recognized as one of the IBM's ten top leading inventors in 1999.