# Indexed Sequential Data Broadcasting in Wireless Mobile Computing

Ming-Syan Chen*, Philip S. Yu† and Kun-Lung Wu†

Electrical Engineering Department*
National Taiwan University
Taipei, Taiwan, ROC
email: mschen@cc.ee.ntu.edu.tw

IBM Thomas J. Watson Research Ctr.†
P.O.Box 704
Yorktown, NY 10598
email: {psyu, klwu}@watson.ibm.com

**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. In this paper, we explore the issue of indexing data with skewed access for sequential broadcasting in wireless mobile computing. We 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.*

## 1 Introduction

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 recently been identified as an area with an emerging market [5]. A significant amount of research effort has been elaborated upon exploring several aspects of mobile computing, including energy saving [6] [8], cache management [1], query processing [4] and data allocation. As reported in [3], the potential market for mobile computing applications is estimated to be in billions of dollars annually. 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 small batteries for their operations and are not directly connected to any power source. As a result, energy saving is the key issue to resolve before we can anticipate an even wider acceptability for mobile computers.

In general, a mobile server is expected to concurrently serve a large number of clients. Since the cost of broadcasting is independent of the number of users, periodic broadcasting is the prevalent method for information dissemination in a wireless communication environment. 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. The ratio of power consumption in active mode to that in doze mode is approximately 5000 [6]. 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 for data access under such an indexed broadcasting.



0-8186-7813-5/97 $10.00 

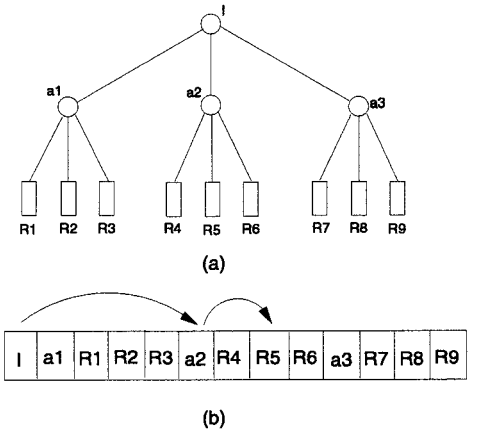


Figure 1: Indexed broadcasting: (a) an example index tree, and (b) index probing scenario to data record $R_5$.

A conventional index tree is given in Figure 1a, and its corresponding broadcasting sequence is given in Figure 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. 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. 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. The absence of research effort in these issues can be explained by the reason that in the conventional file system, the cost of executing an index probe for data access is almost negligible 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 very 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 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 Huffman code [7], 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, 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 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. This paper concludes with Section 4.

## 2 Preliminaries

In this paper, a mobile client is assumed to use selective tuning to listen to indexed sequential data broadcasting. 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 [6], 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. 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. Listening to a channel requires the client to be in the active mode [6]. Hence, tuning time determines the battery consumption. Access time further consists of two components: *probe wait* and *bcast wait*, where probe wait is the time to get the first index and bcast wait is time duration from the point the first index is reached to the point the required record is obtained. For ease of discussion, our interest in this study starts from the point that the first index is captured. Explicitly, we shall explore 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.

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 Figure 1a.

## 3 Index Allocation for Skewed Data Access

### 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 Huffman code, is described below, where $n$ is the number of total data records and $d$ is the fanout of each index.

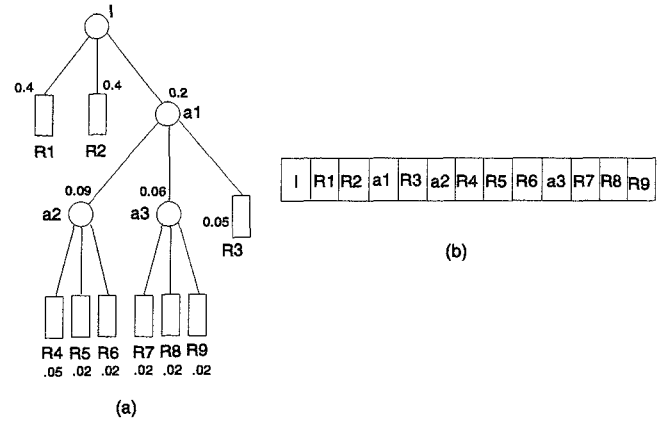


Figure 2: Illustration of an imbalanced index tree: (a) an index tree $T^I_{d=3}$ built by $CF$, and (b) a corresponding data broadcasting sequence.

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.

To facilitate our presentation, the index tree in Figure 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 Figure 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

| Data record | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 |
|---|---|---|---|---|---|---|---|---|---|
| $Pr(R_i)$ | .4 | .4 | .05 | .05 | .02 | .02 | .02 | .02 | .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 |

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 $T^I_{d=3}$.

reduced from 2 (i.e., using the index tree $T^B_{d=3}$ in Figure 1a) to 1.5 (i.e., using the index tree $T^I_{d=3}$ in Figure 2a), thus reducing the average power consumption. A corresponding broadcasting sequence is given in Figure 2b. Consequently, by transforming the problem of minimizing the number of index probes to the one of determining the minimal weighted path length for $d$-ary tree [7], it can be seen that algorithm $CF$ is in essence a generalized version of Huffman code algorithm. Theorem 1 below thus follows.

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

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. For example, for the index tree in Figure 2a, $f(a_2)$=$d(a_2)$=3 and the index probe cost to reach data record $R_5$ is $\sum_{a_j\in Path(R_5)} d(a_j)$= $d(I)$+$d(a_1)$+$d(a_2)$= 3+3+3=9.

Given this cost model, 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 Figures 3 and 4, respectively. Let $C(T^I_{d=j})$ be the average cost of index probes for the index tree $T^I_{d=j}$, i.e., $C(T^I_{d=j})$=$\sum_{1\le i\le n} Pr(R_i) \sum_{a_j\in Path(R_i)} d(a_j)$ for

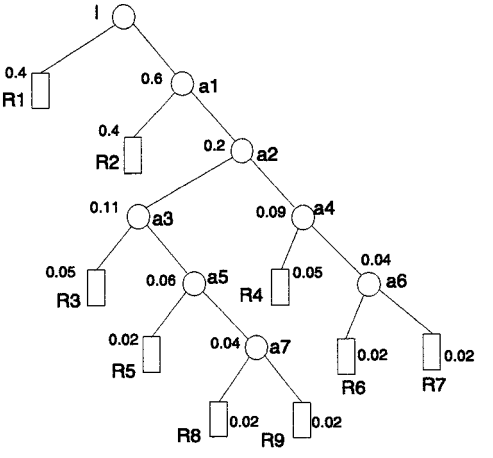


Figure 3: An index tree $T^I_{d=2}$ built by algorithm $CF$.

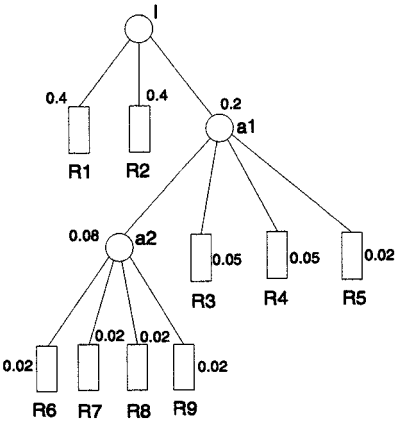


Figure 4: An index tree $T^I_{d=4}$ built by algorithm $CF$.

the index tree $T^I_{d=j}$. It can be obtained from Table 2 that $C(T^I_{d=3})$=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, showing that there is no monotonic relationship, neither increasing nor decreasing, for the values of $C(T^I_{d=j})$ 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 subsection the approach of using vari-

ant index fanouts to 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 [2]. In view of this, we shall develop in the following an efficient heuristic algorithm $VF$ to build an index tree with variant fanouts.

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. Figure 5 shows the scenario of grouping a set of nodes and moving them to a lower level. It can be seen from Figure 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 $h_{i+1}$, $h_{i+2}$, ..., 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 $h_1$, $h_2$, ..., 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, $h_1$, $h_2$, ..., $h_m$, which are sorted according to descending order of $Pr(h_j)$, $1 \leq j \leq m$, i.e., $Pr(h_j) \geq Pr(h_k)$ iff $j \leq k$. Then, the average cost of index probes can be reduced by grouping nodes $h_{i+1}$, $h_{i+2}$, ..., $h_m$ and attaching them under a new child node if and only if

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

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.

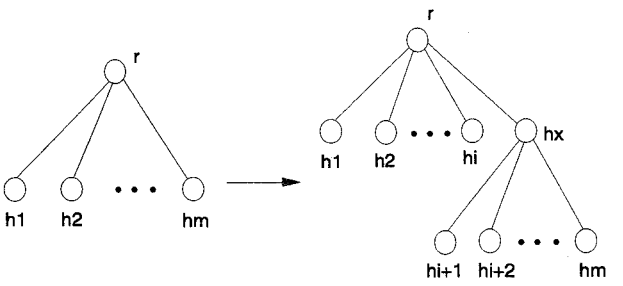


Figure 5: Group a set of nodes and move them to a lower level to reduce the average index probe cost.

Algorithm $VF$:

**Step 1:** Assume that $R_1$, $R_2$, ..., $R_n$ 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* ($R_1$, $R_2$, ..., $R_n$).

**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 Figure 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.

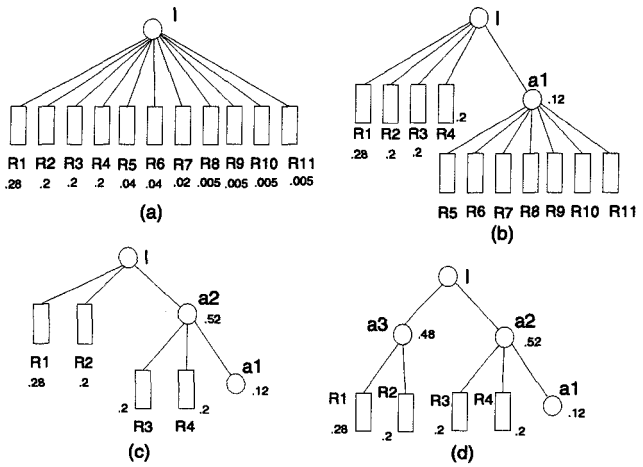


Figure 6: Illustration of algorithm $VF$.

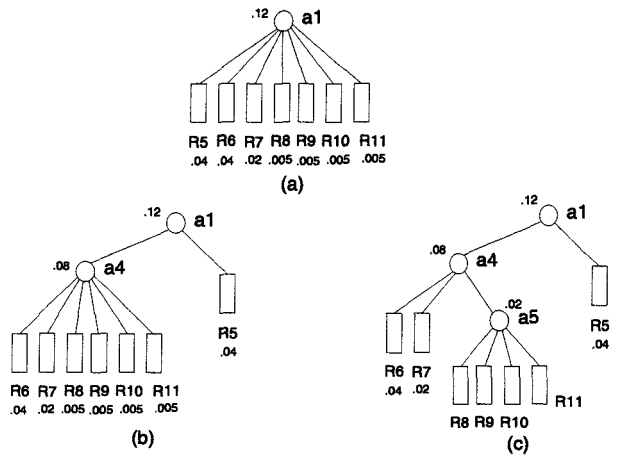


Figure 7: Further partitions of nodes under index $a_1$.

Procedure *Partition* $(h_1, h_2, ..., h_m)$:

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

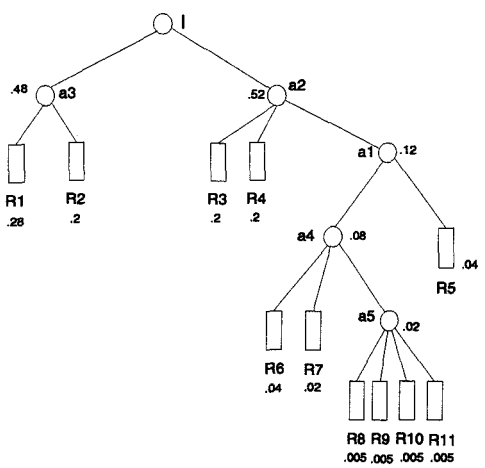


Figure 8: The resulting index tree $T_V^I$ (with variant fanouts) from algorithm $VF$.

For example, consider the profile in Table 3. The initial index tree configuration in shown in Figure 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 $R_5$, $R_6$,..., and $R_{11}$ together and move them to the next level, resulting in the configuration shown in Figure 6b. Nodes $R_5$, $R_6$,..., and $R_{11}$ under the index node $a_1$ are then partitioned recursively as shown in Figure 7. Also, in the original level, nodes $R_5$, $R_6$,..., 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 Figure 6c, which in turn leads to Figure 6d. Consequently, by combining Figures 6 and 7, we obtain the resulting index tree $T_V^I$ in Figure 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 Figure 9, for the same data profile. It can be verified from Figure 9 and Table 3 that $C(T_{d=2}^I) = \sum_{1 \le i \le 11} Pr(R_i) \sum_{a_j \in Path(R_i)} d(a_j) = 5.2$ and $C(T_{d=3}^I) = 5.625$, both larger than $C(T_V^I) = 5.08$, showing that the index tree with variant fanouts obtained by $VF$ requires a smaller average index probe cost than optimal fixed fanout index trees.

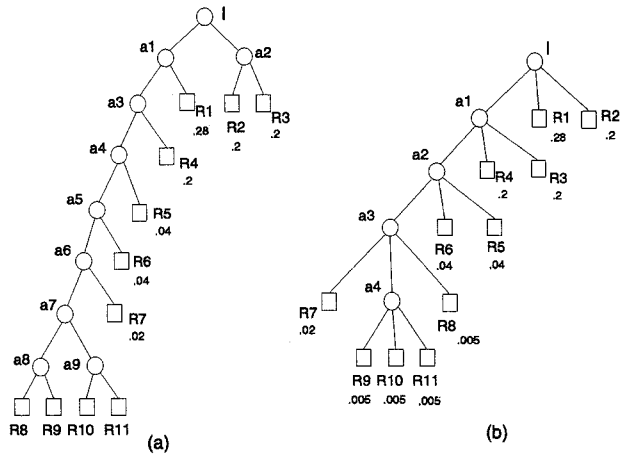


Figure 9: (a) $T^I_{d=2}$ and (b) $T^I_{d=3}$ for the profile in Table 3.

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:

$$y(i) = (g(m) - g(i+1))$$

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

## 4 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 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 $CF$. Examples and remarks were given to illustrate our results.

## Acknowledgement

M.-S. Chen is in part supported by National Science Council, Taiwan, ROC, Contract No. NSC 86-2621-E-002-023-T.

## References

[1] D. Barbara and T. Imielinski. Sleepers and Workaholics: Caching Strategies in Mobile Environments. *Proceedings of ACM SIGMOD, Minneapolis, MN*, pages 1–12, May, 1994.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, CA, 1979.

[3] Y. Huang, P. Sistla, and O. Wolfson. Data Replication for Mobile Computers. *Proceedings of ACM SIGMOD, Minneapolis, MN*, pages 13–24, May, 1994.

[4] T. Imielinski and B. R. Badrinath. Querying in Highly Mobile Distributed Environments. *Proceedings of the 18th International Conference on Very Large Data Bases*, pages 41–52, August 1992.

[5] T. Imielinski and B. R. Badrinath. Wireless mobile computing: Challenges in data management. *Communications of the ACM*, 37(10):19–28, 1994.

[6] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Energy Efficient Indexing on Air. *Proceedings of ACM SIGMOD, Minneapolis, MN*, pages 25–36, May, 1994.

[7] D. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms.* Addison-Wesley, MA, 1973.

[8] S. Sheng, A. Chandrasekaran, and R. W. Broderson. A portable multimedia terminal for personal communications. *IEEE Communications Magazine*, pages 64–75, 1992.

| Data record | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 |
|---|---|---|---|---|---|---|---|---|---|
| $Pr(R_i)$ | .4 | .4 | .05 | .05 | .02 | .02 | .02 | .02 | .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 |

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 | R10 | R11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $Pr(R_i)$ | .28 | .2 | .2 | .2 | .04 | .04 | .02 | .005 | .005 | .005 | .005 |
| $\sum_{a_j \in Path(R_i)} d(a_j)$ in $T^I_V$ | 2 | 4 | 8 | 8 | 10 | 10 | 10 | 12 | 12 | 12 | 12 |
| $\sum_{a_j \in Path(R_i)} d(a_j)$ in $T^I_{d=2}$ | 4 | 4 | 4 | 6 | 8 | 10 | 12 | 16 | 16 | 16 | 16 |
| $\sum_{a_j \in Path(R_i)} d(a_j)$ in $T^I_{d=3}$ | 3 | 3 | 6 | 6 | 9 | 9 | 12 | 12 | 15 | 15 | 15 |

Table 3: The index probe cost required to reach each record by different index trees.

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $(10-i)\sum_{1<j<i} Pr(h_j)$ | 9*.28 | 8*.48 | 7*.68 | 6*.88 | 5*.92 | 4*.96 | 3*.98 | 2*.985 | .99 |
| $\sum_{i+1<j<11} Pr(h_j)$ | .72 | .52 | .32 | .12 | .08 | .04 | .02 | .015 | .01 |
| $y(i)$ | 1.8 | 3.32 | 4.44 | 5.16 | 4.52 | 3.8 | 2.92 | 1.955 | 0.98 |

Table 4: Determining the set of nodes to be grouped together in Figure 6a, where $m = 11$.

| $i$ | 1 | 2 | 3 |
|---|---|---|---|
| $(4-i)\sum_{1<j<i} Pr(h_j)$ | 3*.28 | 2*.48 | .68 |
| $\sum_{i+1<j<5} Pr(h_j)$ | .72 | .52 | .32 |
| $y(i)$ | .12 | .44 | .32 |

Table 5: Determining the set of nodes to be grouped together in Figure 6b, where $m = 5$.