Generalized Indexing for Energy-Efficient Access to Partially Ordered Broadcast Data in Wireless Networks*

Dimitrios Katsaros[1,2] Nikos Dimokas[1] Yannis Manolopoulos[1]
[1]Informatics Department, Aristotle University, Thessaloniki, Greece
[2]Computer & Communications Engineering Department, University of Thessaly, Volos, Greece
{dimitris,dimokas,manolopo}@skyblue.csd.auth.gr

Abstract

Energy conservation and access efficiency are two fundamental though competing goals in broadcast wireless networks. To tackle the energy penalty from sequential searching, the interleaving of index with data items has been proposed. Although, quite important contributions exist on providing broadcast indexes, they have one or more of the following problems. Firstly, all of them assume total ordering among broadcast data, and none considers the more general case of partial ordering. Secondly, they are balanced structures, which does not fit the "linear (one-dimensional) structure" of the wireless medium, in which imbalanced structures may offer significant advantages. Thirdly, they do not take into account the skewness in the access pattern, which prohibits larger performance gains to be reaped. Finally, they require all index items to be of equal size, which may not always give the optimal performance. To cope with all these problems, we introduce a new imbalanced tree-structured index. The new index is shown to be a generalization of two previously proposed high-performance indexes, and it introduces for the first time the problem of indexing partially ordered broadcast data. We present an experimental analysis of the proposed method, contrasting it with competing techniques. The analysis exhibits the efficiency of the proposed index in reducing the energy consumption without noticeably worsening the access latency.

Keywords*: Energy saving, skewed access, indexing, sensor networks, ad hoc networks, wireless networks.*

1 Introduction

Consider the following scenario from sensor network applications, where a node (assumed to be energy-rich, due to its special duties) with various sensing capabilities (e.g., temperature, humidity, pressure, carbon-dioxide concentration), is able to sense the environment at regular time intervals (different for each measured quantity). These measurements are broadcast and collected by surrounding energy-starving nodes/sensors, which implement various application protocols, dealing with aspects of temperature only, of gas only, of concentrations only, etc., or with various combinations of them. Such an application scenario could be built on the basis of the architecture described in [4]. Clearly, ordering of the measurements for the same "quantity" (e.g., temperature, pressure) across time is meaningful, but ordering of measurements of different "quantities" for the same time instance does not make sense. The surrounding application-oriented, energy-starving nodes serving the different applications, will require energy-efficient access to the broadcast information (and not continuous tuning to the broadcast channel). Though, due to the different preferences of the running applications for the measured quantities, the access pattern of the sensors to the disseminated information will be heterogeneous, i.e., skewed.

In a second possible scenario, encountered in a cellular wireless network (e.g., a PCS), resource-constraint mobile units within a wireless cell, retrieve information from a relational database, whose contents are broadcast by a base station serving the cell. The recent Smart Personal Objects Technology (SPOT) by Microsoft proves the industrial interest in such kind of services and exhibits their feasibility. In the general case, the information pieces consist of "projections" (i.e., columns) of relational table rows, for which the *row-key* is able to differentiate only among rows, and not between columns of the same row. Evidently, the mobile users are interested in different combination of rows and columns, and as it is common in realistic situations, the resulting access pattern is skewed.

Finally, in a third scenario inspired from a modern automated battlefield where an ad hoc network is deployed, a power-rich node e.g., an Unmanned Ground Vehicle or Unmanned Airborne Vehicle, is designed to supply intelligence and other tactical operations information to allied troops. It

*Research supported by a $\Gamma.\Gamma.E.T.$ grant in the context of the project "Data Management in Mobile Ad Hoc Networks" funded by $\Pi\Upsilon\Theta A\Gamma OPA\Sigma$ II national research program.

10th International Database Engineering and Applications Symposium (IDEAS'06)
0-7695-2577-6/06 $20.00

broadcasts reports to power-starving mobile units carried, for instance, by individual (or small groups of) soldiers, or by small unmanned exploring vehicles. Apparently, not all transmitted information pieces is of interest to all receiving units and moreover, it may be the case where the "descriptive key" of the information pieces does not impose a global ordering among the pieces.[1]

In these application scenarios, it is evident that:

- The information "consumers" need to retrieve the data as quick as possible (i.e., with small *access latency*, which is the time elapsed between when the need for a datum arises in a node and the moment the node gets that datum from the channel).
- The consumers are energy-starving nodes; therefore they should refrain from continuously monitoring the broadcast channel (i.e., pursuit a small *tuning time*, which is the amount of time a node spends while monitoring the channel).
- There is no global ordering among the data, but only *partial ordering*.
- The access pattern to the broadcast information is *skewed*; some data are more "hot" than other.

Access efficiency is a common objective in many systems, (e.g., databases), but energy conservation is a vital goal in wireless networks for prolonging the longevity of the sensor network or for guaranteeing as much power-independence as possible for the mobile hosts. To achieve energy savings, mobile nodes support two generic modes of operation, the *active* mode, which is a fully operational state, and the *doze* mode, which is a power saving state. The ratio of energy consumption between the two modes is usually an order of magnitude [22]. Similarly, sensor nodes can be in one of three *active* states – transmit, receive, idle – or in *sleep* state; a sensor in the sleep state consumes 7–20 times less energy than when it is in the idle state [5].

Clearly, when the data are broadcast in the channel the node has no alternative, but continuously monitor the channel until it receives them, which results in huge energy penalty. To allow the nodes to remain in the doze/sleep mode so as to save energy, we must incorporate indexing information within or before the broadcast data. Using this *air indexing* [11], which includes information about arrival times of data, the nodes need not stay tuned to the channel all the time, but they can "wake up" at specific instances, when data useful to the node are to be transmitted.

Disk-based indexing techniques is thoroughly investigated in the database literature. Some of these techniques have been adapted to the wireless case, e.g., hashing [10, 23], B^+-tree [11, 25, 27], with different tradeoffs between access and tuning time performance, but the B^+-tree based methods exhibit the best overall performance [25].

[1]The terms (*information*) *piece*, *datum* and *data item* are used interchangeably.

However, all existing indexing schemes (cf. Subsection 2.2) are designed with the assumption that there exists a global ordering among the indexed data. The only exception is the index tree proposed in [3], which assumes no ordering at all among the indexed data. Unfortunately, this scheme is not an indexing method at all, because it requires scanning the whole tree in order to discover an item, since the tree provides no guidance (i.e., branch following). In addition, most of the indexing schemes [11, 23, 25, 27] are tailored to uniform data access, thus ignoring the possibility of reaping large performance gains (in terms of access time) from the skewed access pattern. A detailed discussion of the relevant work along with the shortcomings of each method will be presented in Subsection 2.2.

We propose a novel tree-structured index, named the *Partial Ordering Broadcast Index,* $\mathcal{POBI}$ to be interleaved with or precede broadcast data, which minimizes the energy expenditure at minimal access time aggravation, and it is tailored to realistic skewed access patterns. The novelty of the proposed structure is the fact that the index can handle partial ordering among the transmitted data, which is a feature that is not present in any previously presented index. The key point is that the proposed index is a natural generalization of two state-of-the-art indexing schemes, i.e., the *Variant Fanout*, VF [3] and k*-ary Alphabetic Tree*, kAT [21] indexes. A performance analysis of the index in terms of the access latency and tuning time is provided. A wide range of experiments are conducted to compare the proposed index with the state-of-the-art relevant air-indexing schemes.

The rest of this paper is organized as follows. Section 2 introduces useful concepts and possible assumptions and it surveys the research work related to indexing broadcast data. Section 3 presents the formulation of our problem and Section 4 presents the construction method of the novel broadcast index; we compare the proposed index with the existing schemes in Section 5. Finally, the paper is concluded in Section 6.

2 Background

2.1 Preliminaries

We consider a generic data dissemination system which uses broadcasts to send information to clients. There exists a resource-rich node, we name it the *server*, which broadcasts the information through a broadcast channel. Each information piece can be identified by a key value. This key value does not impose a global ordering among the data, but only partial order. The nodes are resource-constraint devices, and they can be, for instance, sensor nodes of a wireless sensor network or mobile hosts roaming inside the coverage area of a Personal Communications System. We assume that the server is able to estimate the popularity of each information piece, e.g., using subscriptions. In order

0-7695-2577-6/06 $20.00

to retrieve the required broadcast data, the nodes tune into the broadcast channel and retrieve the packets containing indexing information until they get the information about the arrival time of the sought datum. This *selective tuning mechanism* achieves the energy savings.

We assume here a single broadcast channel. Our proposed index $\mathcal{POBI}$ though, can be adapted with minimal design changes to the existence of multiple channels.

Although not critical for our study, we assume a *flat broadcast*, where each datum appears exactly once, instead of *skewed broadcast* [1], where some items may appear more than once. The proposed index $\mathcal{POBI}$ can straightforwardly be extended to support replicated items. Apart from data replication, a number of other client-side techniques can also be used to reduce the access time, like data caching [13], prefetching [14], but all these are orthogonal to our work and will not be considered.

Here, we assume that a server broadcasts n equi-sized data records, each denoted as R_i $(1 \leq i \leq n)$ and with access probability $Pr(R_i)$, where $\sum_i Pr(R_i) = 1$. Let $I_{pb}(R_i)$ be the number of index probes to reach R_i in the index tree and $d(a_i)$ be the fanout of a node a_i of the index tree. We denote with *Path*(R_i) the set of index nodes from the root of the tree to R_i. Similarly to [3], we define the average cost of any index tree as:

$$\sum_{1 \leq i \leq n} Pr(R_i) * I_{pb}(R_i), \tag{1}$$

where the cost of index probing is estimated as:

$$I_{pb}(R_i) = \sum_{a_j \in Path(R_i)} d(a_j). \tag{2}$$

Although the definition of alternative cost models is possible, this model is the most realistic in the literature. Having defined the cost formula, we can easily see why non-uniform access pattern combined with unbalanced tree structures provides opportunities for gains. Initially, consider four data items with uniform access patterns, i.e., $Pr(R_1) = Pr(R_2) = Pr(R_3) = Pr(R_4) = 0.25$ (ordered as $Pr(R_1) < Pr(R_2) < Pr(R_3) < Pr(R_4)$) and suppose we build a balanced tree like the one described in [11] with fanout $d(a_i) = 2$. Then, the resulting tree will be the one illustrated in the left part of Figure 1 and its cost will be the optimal one and equal to $\sum_{1 \leq i \leq 4} Pr(R_i) * I_{pb}(R_i) = \sum_{1 \leq i \leq 4} 0.25 * 4 = 4$. Suppose now, that the access pattern is skewed, and the probabilities are $Pr(R_1) = 0.40$, $Pr(R_2) = Pr(R_3) = Pr(R_4) = 0.20$. Then, the tree in the left part of Figure 1 would result in a cost equal to 4, whereas the tree in the right part of Figure 1 would result in a cost equal to 3.8, reaping gains up to 5% even for this tiny example.

Our target is to create an index tree minimizing the cost of Equation 1, under the assumptions of non-uniform access pattern and partial data ordering. The complete problem definition will be given in Section 4.

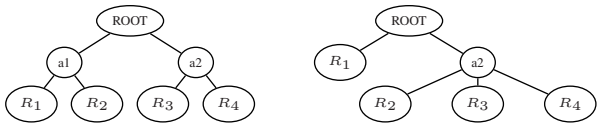

Figure 1. Opportunities for gains by exploiting access skewness and tree imbalancing.

2.2 Related work

Disk-based indexing (e.g., B$^+$-trees, Hashing, R$^+$-trees, Quadtrees, Skip Lists) for traditional as well as for advanced applications is a thoroughly investigated area during the past years. An adaptation of the idea of B$^+$-tree indexing in wireless environments was first described in [11], where instead of the disk addresses the leaves of the B$^+$-tree store the arrival time of each datum in the broadcast channel. Similarly, an adaptation of the traditional hash-based indexing technique in wireless environments was earlier described in [10], and later was generalized in [23]. Hybrids between the two approaches are described in [20, 25] and application of signature trees as indexing methods in reported in [15], which of course can support only equality queries. Adaptation of such indexing schemes (e.g., B$^+$-tree) to work in multiple broadcast channels are described in [8, 26]. They do not propose new schemes but simply different allocation methods for the nodes of the indexing tree. In all these works it is assumed that: a) there is a global ordering among the transmitted data, and b) the access pattern is uniform, that is, the access probability is the same for all data, which is quite unrealistic.

Deviating from the uniform access probability assumption, several works considered the effect of access skew on the design of indexing schemes. A new scheme is proposed in [21], which is a k-ary version of the basic binary Alphabetic Tree [9] over the data, whereas [24, 26] adapted the indexing method of [11] to deal with non-uniformity in access. Various methods were based on the construction of a binary or k-ary Alphabetic Tree to develop indexing schemes for multiple broadcast channels [12, 17]. These methods do not provide new types of tree-structured indexes, but rather a new allocation method for the tree-structured method of Alphabetic trees to the multiple channels. All these works [12, 17, 21, 24, 26] assume that: a) there is a global ordering among the transmitted data.

There is only one work [3], which deviated from both the uniformity and global ordering assumptions. Though, it assumed *no ordering at all* for the data, therefore turning the proposed indexing not to be a *searching structure* at all.

For the case of wireless sensor networks, since the majority of research has focused for the moment on topics like

routing, clustering, sleep scheduling, localization, medium access control, the issue of indexing has received much less attention and the literature has solely developed distributed indexes that reside on the sensor nodes and are not broadcast. These indexes comprise (in one form or another) adaptations of the traditional disk-based indexes, with special care to achieve only local (to the extend possible) communication during their creation or maintenance, and small storage overhead. The GHT [19] is based on a (geographic) hashing scheme, DIM [16], DIFS [7] and DIST [18] are based on the quadtree structure, and TSAR [4] is based on Skip Graphs (a generalization of Skip Lists for distributed environments). None of these indexes is broadcast over wireless channels and they all assume global ordering for the data they index.

3 The Partial Ordering Broadcast Index construction problem

Assume (as stated in subsection 2.1) that we are given a set of equi-sized n data items, each with access probability of $Pr(R_i)$. Also, assume that these n items are distributed (by the nature of the each application) into m bins (or groups) B_j; each bin B_j contains $|B_j|$ items. Without lost of generality, we assume that bin B_1 contains the items $1, \ldots, |B_1|$, bin B_2 contains the items $|B_1|+1, \ldots, |B_1|+|B_2|$, and bin B_l contains the items $\sum_{1\leq x\leq l-1}|B_x|+1, \ldots, \sum_{1\leq x\leq l-1}|B_x|+|B_l|$, where $\sum_{1\leq j\leq m}|B_j| = n$. Moreover, assume that for the items within a bin there is no ordering, whereas each item of bin B_i is "smaller" than each item of bin B_j, iff $i < j$. Then our goal can be formulated in mathematical terms as follows:

Definition 1 (Partial Ordering Broadcast Index). *Given the number n of data items, their access probabilities $Pr(R_i)$, the number m of bins (groups) B_j and a membership function (specifying which item belongs to which bin), our goal is to construct an indexing tree minimizing the cost of Equation 1, such that in the tree we build, the item x precedes item y in an inorder traversal of the tree, if $x \in B_i$ and $y \in B_j$ for $i < j$. The order of items within each bin is arbitrary.*

3.1 Generalizing earlier indexes

Clearly, our problem definition embraces the problems defined in [21] and [3]. If we consider that there is only one bin, i.e., $m = 1$, then our problem reduces to that in [3], since there will be no ordering at all among the data, and it can be solved as a formulation of (binary or k-ary) Huffman tree. On the other hand, if there are as many bins as the number of items ($m = n$) and each bin accommodates only one item, then our problem reduces to that in [21], since a global ordering among all data will be imposed, and thus the problem can be solved as a formulation of the (binary or k-ary) Alphabetic tree.

4 The Partial Ordering Broadcast Index

To deal with the aforementioned problem, which defines the partial ordering broadcast index, we firstly attempt to gain some insights into it. Apparently, a brute force approach is to generate every possible permutation of the n items (respecting the group membership and inter-group ordering) and construct an alphabetic tree over them, selecting the one with the least cost. It is clear that such a method of attack is inefficient, since we have to examine exponentially many permutations, roughly, the product of each group's possible permutations. Indeed, we can easily see that we can not design a polynomial-time algorithm like that of [6], to solve our problem. Therefore, the exploitation of smart approximations with small execution time is preferable.

A cheap solution is to select a random ordering for the items inside each group and then apply a binary or k-ary alphabetic tree construction algorithm to all the items. We can prove no analytic performance results for such a solution, but we can do better than this. We call this baseline naive scheme, as $kATr$, where parameter k reveals the fanout of each node (for a binary tree $k = 2$).

Can the ordering of items in each group provide some benefits? It can be proven that for the special case where all groups, but the first and last group, contain only one item, then the lowest cost $\mathcal{POBI}$ can be obtained if we order the items of the first group in non-ascending order and the items of the last group in non-descending order. But in the case that several groups, not only the first and last, contain more than one item, then such an ordering can not lead to the optimal solution. Using these orderings though, we can design some approximations to our problem. In particular, an heuristic could be obtained by sorting the items of each group in non-descending order and then constructing an alphabetic tree; we call it $kATi$. Similarly, by sorting the items of each group in non-ascending order, we get another heuristic, the $kATd$.

The aforementioned approximations are relatively straightforward making use of some ordering, based on popularity among the items of each group, trying to push the less popular items deeper into the resulting tree. Their drawback though is that the rely on the alphabetic tree construction algorithm to form the final tree. Departing from these naive heuristics, we adopt a more structured way, with the same objective of trying to push the less popular items of each group deeper into the resulting broadcast search tree. This structured approach consists of creating subtrees, each subtree corresponding to one group. Then, treating each subtree as a single node with the tree's cost as the node's weight, we apply an alphabetic tree construction algorithm.

The main issue in this approach is to devise a subtree construction algorithm. We propose the following three generic strategies: a) "place the most popular item at the root of the tree, then proceed similarly on the branches of

the root", b) "choose the root so as to equalize the total weight of the branches", and c) "construct a Huffman tree with variant fanout over the items of each group". We name these approximations as $MostPop$, $EqWeig$ and $\mathcal{POBI}$, respectively. The first two strategies are simple to understand. We explain in a little more detail only the third one.

This strategy works similarly on all groups. It starts from an initial node, say x which has as its children all items of the group, i.e., $c_1, c_2, \ldots c_y$. Initially, it sorts in non-ascending popularity the items of the group. Then, if it finds a z, such that $(y - z - 1) * \sum_{1 \leq j \leq z} Pr(c_j) > \sum_{z+1 \leq j \leq y} Pr(c_j)$, it creates a new node, say nx as father of the nodes $c_{z+1}, \ldots, c_y$ and makes this new node as child of the node x. This partitioning is applied recursively to both x and nx, until no further partitioning is possible. This procedure is tailored to minimizing the cost of Equation 1.

4.1 Directing the search in the subtrees of each group

In order to characterize a tree structure as a search tree, it must posses a distinct feature, i.e., the internal nodes of the tree must provide guidance as where to continue the search, while seeking for a particular item. The tree structures, like B^+-tree [11, 25], and alphabetic tree [21] do posses this feature and thus they are characterized as search tree. On the other hand, the VF tree [3] *is not* a search tree, because, in an inorder traversal of the tree, the items are not retrieved in the sorting (lexicographic or numeric) order; therefore there is no way to use the internal nodes of this tree to prune parts of the tree while searching.

Our developed tree structure faces a similar problem, though to a more limited extent, because we only need to provide guidance for the items of each group. Suppose we are seeking for an item i belonging to group g. We can easily discover the subtree, which corresponds to group g, by comparing item i to other items not belonging to group g, since there is an ordering between i and these items. At the root of the subtree corresponding to group g, we must make a decision about which branch to follow, that will direct us to item i. Each node is equipped with a Bloom filter [2], that provides this facility. Thus, we are able to bypass the lack of ordering between the items of a group, at the expense of a very small false-drop probability, i.e., some cases where we have to follow more than one branch, which though can be controlled to become really insignificant.

5 Performance Evaluation

For the evaluation of the proposed algorithms, we developed a system that simulates an environment where a number of nodes (clients) access the data served by a server through a broadcast channel. Each node has a cache to store previously accessed data and selects the next data to access based on its profile. Whenever a node needs an item which is not stored locally, it tunes to the broadcast channel reading the index information and alternating between sleep and active mode, until it gets the required information.

We implemented a Zipfian model for node request distribution, for group sizes distribution, for item popularity distribution and group popularity distributions, i.e.,

$$p_x = \frac{(1/x)^\theta}{\sum_{x=1}^{y}(1/x)^\theta} \quad 1 \leq x \leq n \text{ (or } 1 \leq x \leq m) \qquad (3)$$

where θ controls the *skewness* of the distribution. For $\theta = 0$ the Zipfian reduces to the uniform distribution, whereas larger values of θ derive increasingly skewer distributions. Using this formula, we can derive relative popularities for items and groups and relative sizes for the groups.

The formation of groups and item popularities were generated roughly as follows. Suppose we have decided to generate n items to be assigned to m groups. Firstly, we decide the relative sizes of the groups using the Zipfian distribution (Equation 3); setting small values for θ all groups have similar sizes, whereas for larger θ values some groups have significantly larger number of items. Then, we assign to each group a relative popularity, using again the zipfian law. Depending on the combination, we can generate large groups with large popularity, small groups with large popularity, equi-sized groups with roughly equal popularity and so on. Then in a third invocation of the Zipf's law, this popularity is distributed to the group's items. The preference of users to items are decided again with the Zipfian law.

We performed a large number of experiments to assess the impact of the volume of data, of the access distribution of the data, of the client cache, of the very interesting tradeoffs involved in the use of Bloom filters, etc. For the interest of space, in this article we take up only with the analytic assessment of the proposed tree structures as they are defined by Equation 1. In particular, we will present the performance of the algorithms with varying number of items, with varying number of groups, with varying popularity distribution of the groups and with varying relative size (in terms of items) of the groups; all these for a couple of combinations for the default parameters. In Table 1 we present the major symbols used in the simulation, along with a brief explanation for it and its default value. We use the notation nAgBgsCgpD in the title of the plots to denote the parameters, where each of the symbols A, B, C or D is replaced either from numeric value or from an underscore to denote that it is varying. For instance, the notation n500g10gs01gp_ means that we are investigating various values for the relative group popularity in a collection of 500 items divided into 10 groups, where the relative skewness in group sizes is 0.1.

Since this article introduces for the first time the problem of indexing partially ordered data for wireless broadcasts, there are no relevant competing methods in the lit-

0-7695-2577-6/06 $20.00

Var	Meaning	Default
n	# of database items	500
g	# of groups	10
gs	skew in group size	0.1 ($0.0 \rightarrow$ equi-sized)
gp	skew in group popularity	0.1 ($0.0 \rightarrow$ equi-popular)

Table 1. Simulation parameters.

erature to compare with. Though we have already devised some straightforward extensions of existing tree structures, which can serve the role of the competitor methods. We developed a straightforward extension of the VF tree [3], with the same name, to take into account the partial ordering of items. Also, we implemented a straightforward extension of the original k-ary Alphabetic tree, by employing a random ordering of the items inside each group and denote this variant as kAT, as well, but it coincides with $kATr$; consequently we do not expect this variant to show good performance. Finally, we implemented our approximation schemes, namely $kATi$, $kATd$, $MostPop$, $EqWeig$ and $\mathcal{POBI}$. In the plots of the following subsection, we exclude from the presentation the $EqWeig$ method, because it exhibited the same performance as that of $MostPop$.

5.1 Impact of the number of items

The first experiment aimed to investigate the impact of the number of items on the relative performance of the algorithms. We varied the number of items from 100 to 1000 and investigated the average index cost incurred by the algorithms along two dimensions, i.e., relative group size and relative group popularity, considering equal and quite skewed sizes for the groups and also considering equal and quite skewed popularities; thus we came up with four combinations. The resulting graphs are illustrated in Figure 2.

The general trend is the increase of the average index cost with increasing number of items. This is due to the fact that the larger the number of items is, the deeper the resulting tree is. The average index cost increases in a logarithmic fashion, since the resulting tree is quite bushy, and not a degenerate left (or right) deep tree. Regarding the performance of each algorithm, the first observation is that the kAT method is the worst of all, as expected, since it makes no smart decisions in the construction of the search tree. The performance of the methods, which are based on a kind of sorting of the items of each group, before constructing the final alphabetic tree (i.e., $kATi$, $kATd$ and $MostPop$) is almost equal (indistinguishable in the plots), but significantly better than that of kAT.

The most effective algorithms are VF and $\mathcal{POBI}$, with the latter to be the champion method outperforming the former in all cases. It is interesting to node that, ignoring some statistical error, the performance of VF deteriorates compared to that of $\mathcal{POBI}$, with increasing number of items. This is obviously due to the better decisions taken by $\mathcal{POBI}$, which exploits intra-groups item relationships.

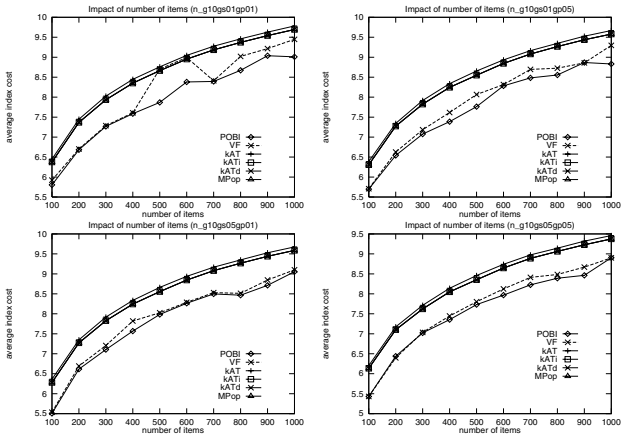

Figure 2. Impact of the number of items on tree performance.

5.2 Impact of the number of groups

We investigated the impact of the number of groups on the relative performance of the algorithms. We varied the number of groups from 10 to 50 and investigated the average index cost incurred by the algorithms along two dimensions again, i.e., relative group size and relative group popularity, considering equal and quite skewed sizes for the groups and also considering equal and quite skewed popularities; thus we came up with four combinations. The resulting graphs are illustrated in Figure 3.

The general trend here is not similar for all four graphs; it is apparent although not very intense. In the upper two plots of the figure, which represent the case where the groups are equi-sized, where observe a gradual increase in the average index cost, with increasing number of groups. This is due to the fact that the larger the number of items is, the deeper the resulting tree is. But, in the case of the lower two plots, which represent the case where there is a skewness in the groups' sizes, we can observe that the average index cost is relative stable and thus robust to the increasing group number. This behaviour has its origin in the fact that the skewness in group sizes results in a few groups collecting the largest part of probability mass; thus these few groups are responsible for incurring the largest part of index cost, which cost, in its turn, remains relatively unaffected by the large number of groups.

Regarding the performance of each algorithm, the comment for the kAT, $kATi$, $kATd$, $MostPop$ are analogous with those when we examined the impact of the number of items. The interest here lies is in the relative performance of VF and $\mathcal{POBI}$. The general trend is that VF is less effective than $\mathcal{POBI}$, when the groups are relatively equi-sized, and it achieves to close its performance lag, only the

0-7695-2577-6/06 $20.00

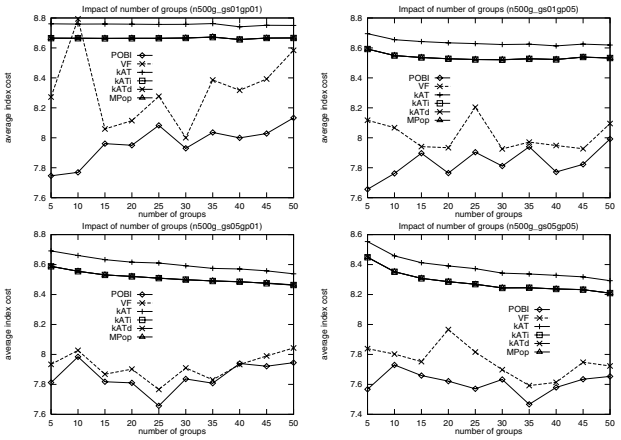

Figure 3. Impact of the number of groups on tree performance.

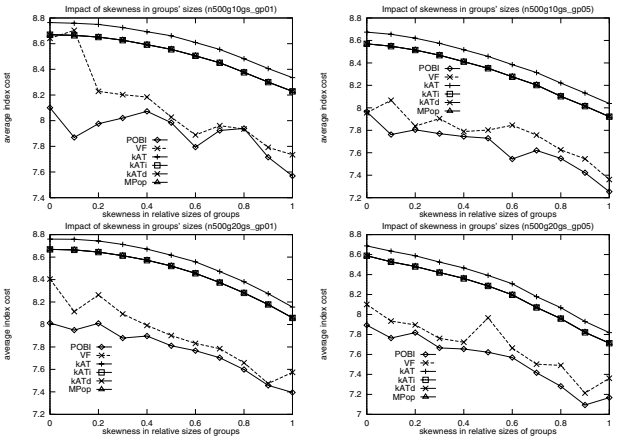

Figure 4. Impact of the relative groups's sizes on tree performance.

skewness in groups' sizes or the skewness in groups' popularity becomes significant. In these cases the largest part of probability mass is concentrated in a few groups (items) and thus the erroneous decisions in the tree construction procedure by VF become less significant, because it can handle relatively easily a few but very popular items. On the other hand, $\mathcal{POBI}$'s performance is not really affected by the number of groups, except from the case of equi-sized and equi-popular groups, where it naturally exhibits a small increase in the average index cost.

5.3 Impact of the relative group sizes

The third experiment investigated the impact of the relative sizes of groups on the performance of the algorithms. We varied the skewness in groups sizes from 0.0 to 1.0 and investigated the average index cost incurred by the algorithms along two dimensions, i.e., the number of groups and the relative group popularity, considering 10 and 20 groups, and also considering equal and quite skewed popularities; thus we came up with four combinations. The resulting graphs are illustrated in Figure 4.

The general trend is a decrease of the average index cost with increasing skewness in group popularity; this is expected since the largest part of popularity mass in concentrated into fewer groups and thus items, which items are successively placed on the upper parts (near the root) of the resulting search trees. Now we can clearly see that the $\mathcal{POBI}$ index is particularly better than VF for the cases when there is no excessive skewness in the group popularity, but the skewness is moderate or small. The reason has adequately being explained in subsection 5.2. In any case though, it outperforms the VF tree.

5.4 Impact of relative group popularity

Finally, the last experiment aimed to investigate the impact of the relative group popularity on the performance of the methods. The conclusions drawn from this experiment for all methods are consistent with those being observed in the aforementioned plots. $\mathcal{POBI}$ reveals another virtue; for relatively large amount of skewness in group popularity is able to reduce the average index cost in even larger pace, than that for moderate and small skewness. This is evident from the larger slope exhibited by the $\mathcal{POBI}$'s curve for skewness values beyond 0.5.

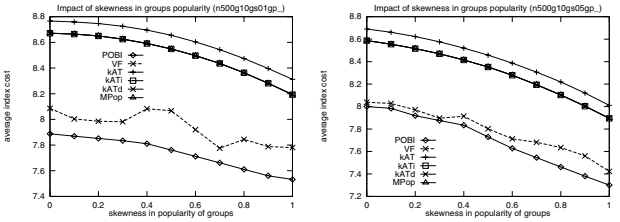

Figure 5. Impact of the relative groups's popularity on tree performance.

6 Summary

This paper investigated for the first time the issue of indexing broadcast information, for the case of partially ordered data. The emerged applications running over wireless ad hoc or wireless sensors are the the main motivation for investigating the case of partially ordered data. Though, usual applications like those developed for the now traditional wireless cellular networks could benefit from the issues discussed in the article.

We defined the problem of indexing partially ordered data and showed that it naturally generalizes two problems proposed earlier in the literature, that lead to the devel-

0-7695-2577-6/06 $20.00

oped of two high performance indexing structures. We provided approximation algorithms to generate the broadcast search trees that attack the defined problem, since providing optimal algorithms require solving exponential number of subproblems. The design of the search structure took into account the skewness in access distribution and thus we proposed an imbalanced structure as solution to the access skewness.

We implemented a simulation environment to investigate the performance of the proposed approximation algorithms and presented detailed experiments that assess the superiority of one of the methods, namely of the $\mathcal{POBI}$ index search tree. $\mathcal{POBI}$ was proven to be the champion algorithm outperforming clearly all other competing methods.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: Data management for asymmetric communications environments. In *Proceedings of the ACM Conference on Management of Data (SIGMOD)*, pages 199–210, 1995.

[2] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[3] M.-S. Chen, K.-L. Wu, and P. Yu. Optimizing index allocation for sequential data broadcasting in wireless mobile computing. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):161–173, 2003.

[4] P. Desnoyers, D. Ganesan, and P. Shenoy. TSAR: A two tier sensor storage architecture using interval skip graphs. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 39–50, 2005.

[5] L. M. Feeney. Energy efficient communication in ad hoc wireless networks. In S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, editors, *Mobile Ad Hoc Networking*, pages 301–327. IEEE/Wiley, 2004.

[6] E. N. Gilbert and E. F. Moore. Variable-length binary encodings. *The Bell System Technical Journal*, 38:933–967, Jul. 1959.

[7] B. Greenstein, S. Ratnasamy, S. Shenker, R. Govindan, and D. Estrin. DIFS: A distributed index for features in sensor networks. *Ad Hoc Networks*, 1(2–3):333–349, 2003.

[8] C.-H. Hsu, G. Lee, and A. L. P. Chen. Index and data allocation on multiple broadcast channels considering data access frequencies. In *Proceedings of the Conference on Mobile Data Management (MDM)*, pages 87–93, 2002.

[9] T. Hu and A. Tucker. Optimal computer search trees and variable-length alphabetical codes. *SIAM Journal on Applied Mathematics*, 21(4):514–532, 1971.

[10] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Power efficient filtering of data on air. In *Proceedings of the Conference on Extending Data Base Technology (EDBT)*, pages 245–258, 1994.

[11] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Data on air: Organization and access. *IEEE Transactions on Knowledge and Data Engineering*, 9(3):353–372, 1997.

[12] S. Jung, B. Lee, and S. Pramanik. A tree-structured index allocation mathod with replication over multiple broadcast channels in wireless environments. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):311–325, 2005.

[13] D. Katsaros and Y. Manolopoulos. Web caching in broadcast mobile wireless environments. *IEEE Internet Computing*, 8(3):37–45, 2004.

[14] D. Katsaros and Y. Manolopoulos. Prediction in wireless networks by Markov chains. *IEEE Wireless Communications*, 2006. to appear.

[15] W.-C. Lee and D. L. Lee. Using signature techniques for information filtering in wireless and mobile environments. *Distributed and Parallel Databases*, 4(3):205–227, 1996.

[16] X. Li, Y.-J. Kim, R. Govindan, and W. Hong. Multidimensional range queries in sensor networks. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 63–75, 2003.

[17] S.-C. Lo and A. L. P. Chen. Optimal index and data allocation in multiple broadcast channels. In *Proceedings of the IEEE Conference on Data Engineering (ICDE)*, pages 293–302, 2000.

[18] A. Meka and A. K. Singh. DIST: A distributed spatio-temporal index structure for sensor networks. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 139–146, 2005.

[19] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with GHT: A geographic hash table. *ACM Mobile Networks and Applications*, 8(4):427–442, 2003.

[20] A. Seifert and J.-J. Hung. FlexInd: A flexible and parameterizable air-indexing scheme for data broadcast systems. In *Proceedings of the Conference on Extending Data Base Technology (EDBT)*, volume 3820 of *Lecture Notes in Computer Science*, pages 902–920, 2006.

[21] N. Shivakumar and S. Venkatasubramanian. Efficient indexing for broadcast based wireless systems. *ACM/Baltzer Mobile Networks and Applications*, 1(4):433–446, 1996.

[22] M. A. Viredaz, L. S. Brakmo, and W. R. Hamburgen. Energy management on handheld devices. *ACM Queue*, 1(7):44–52, 2003.

[23] J. Xu, W.-C. Lee, X. Tang, Q. Gao, and S. Li. An error-resilient and tunable distributed indexing scheme for wireless data broadcast. *IEEE Transactions on Knowledge and Data Engineering*, 18(3):392–404, 2006.

[24] J. Xu and K.-L. Tan. An analysis of selective tuning schemes for nonuniform broadcast. *Data and Knowledge Engineering*, 22(3):319–344, 1997.

[25] X. Yang and A. Bouguettaya. Adaptive data access in broadcast-based wireless environments. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):326–338, 2005.

[26] W. G. Yee and S. B. Navathe. Efficient data access to multi-channel broadcast programs. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 153–160, 2003.

[27] W. G. Yee, S. B. Navathe, E. Omiecinski, and C. Jermaine. Bridging the gap between response time and energy-efficiency in broadcast schedule design. In *International Conference on Extending Data Base Technology (EDBT)*, pages 572–589, 2002.