International Journal of Geo-Information

MDPI

Article

Spatial Air Index Based on Largest Empty Rectangles for Non-Flat Wireless Broadcast in Pervasive Computing

Jun-Hong Shen [1,2,*], Ching-Ta Lu [1,2], Mu-Yen Chen [3] and Chien-Tang Mai [1]

1 Department of Information Communication, Asia University, Taichung 413, Taiwan; lucas1@ms26.hinet.net (C.-T.L.); purelydinkum@gmail.com (C.-T.M.)
2 Department of Medical Research, China Medical University Hospital, China Medical University, Taichung 404, Taiwan
3 Department of Information Management, National Taichung University of Science and Technology, Taichung 404, Taiwan; mychen@gm.nutc.edu.tw
* Correspondence: shenjh@asia.edu.tw; Tel.: +886-4-2332-3456 (ext. 20006)

Academic Editors: Jason C. Hung, Yu-Wei Chan, Neil Y. Yen, Qingguo Zhou and Wolfgang Kainz
Received: 31 August 2016; Accepted: 9 November 2016; Published: 11 November 2016

Abstract: In pervasive computing, location-based services (LBSs) are valuable for mobile clients based on their current locations. LBSs use spatial window queries to enable useful applications for mobile clients. Based on skewed access patterns of mobile clients, non-flat wireless broadcast has been shown to efficiently disseminate spatial objects to mobile clients. In this paper, we consider a scenario in which spatial objects are broadcast to mobile clients over a wireless channel in a non-flat broadcast manner to process window queries. For such a scenario, we propose an efficient spatial air index method to handle window query access in non-flat wireless broadcast environments. The concept of largest empty rectangles is used to avoid unnecessary examination of the broadcast content, thus reducing the processing time for window queries. Simulation results show that the proposed spatial air index method outperforms the existing methods under various settings.

Keywords: location-based services; non-flat wireless broadcast; spatial air index; window queries

1. Introduction

In pervasive computing, location-based services (LBSs) provide valuable services to mobile clients based on their current locations. LBSs delivered via wireless data broadcast can efficiently and simultaneously serve applications to an enormous number of mobile clients [1–7]. Moreover, LBSs enable context-aware systems to provide applications that can react based on the mobile clients' current locations. LBSs rely on spatial window queries to deliver useful applications to mobile clients [8]. Such queries retrieve spatial objects in a fixed rectangular region based on the clients' current locations [1,2]. For example, a user may use his mobile client to issue a query to locate ATM machines within one kilometer of his current location.

Skewed access patterns of mobile clients result in spatial objects being retrieved at relatively high rates of frequency (e.g., well-reviewed restaurants for a specific area). Such "hot" objects should appear more frequently in a broadcast cycle to result in a non-flat broadcast [9], thus reducing the average client waiting time to retrieve data from the wireless channel. In this paper, we address the issue of how to support efficient access for processing window queries in non-flat wireless broadcast systems. In such systems, the antenna-equipped server cyclically broadcasts a set of selected spatial objects, thus forming a broadcast cycle on a wireless channel to generate a broadcast stream, and mobile clients tune in to the channel to retrieve the desired data.

To maximize battery life, mobile devices need to stay in doze mode whenever possible and switch into active mode only when necessary [10–13]. Interleaving spatial indexes with spatial objects could help mobile devices predict the arrival times of spatial objects and their related indexes. This would allow mobile devices to ignore unwanted data, thus reducing energy consumption [14–16]. Access and tuning times are the two main performance measures for spatial air index methods in wireless broadcast systems. Access time represents the amount of time required from the mobile client issuing a query to the point at which all response data are completely retrieved. Tuning time represents the amount of time for which the mobile device is active to retrieve the related indexes and data to process a query.

Previous efforts have developed spatial indexes to support spatial query processing in non-flat wireless broadcast systems. Im and Choi (2012) proposed a multi-leveled air index scheme (*MLAIN*) [17], which adopts a multi-level division for the space to outperform previous methods. However, *MLAIN* only partitions two groups, hot and regular data, and all hot data are repeatedly broadcast on the same frequency in a broadcast cycle. This increases the length of the broadcast cycle, and causes a massive increase in access time. To address this problem, Shen and Jian proposed a skewed spatial index (*SSI*) [18] to further partition hot data into multiple groups, and repeat the hot data in a broadcast cycle with different frequencies.

However, both of these cell-based methods have a serious weakness in terms of query processing. In both methods, to process a window query, mobile devices should examine all cells overlapping with the query region, even though some cells do not contain the relevant objects and can be skipped. This results in unnecessary examination of cells broadcast on the wireless channel, thus increasing access and tuning times. Therefore, considering the skewed access patterns of mobile clients, we propose a spatial air index method based on largest empty rectangles, named *LER*, to efficiently process spatial window queries issued by mobile clients in non-flat wireless broadcast systems. The largest empty rectangle represents the rectangle of maximum size extending from the minimum bounding rectangle containing the objects in the hot cell, without touching any spatial object [19]. If the query region of a window query is contained in the largest empty rectangle of the cell, the relevant objects are definitely located in this cell, and the other cells can be skipped. In this way, mobile devices can quickly finish query processing, shortening access and tuning times, thus minimizing energy consumption.

Our research contributions are summarized as follows.

1. For processing window queries, the information about the largest empty rectangle of the cells is provided in the spatial air index to further bound the search region and skip irrelevant broadcast content, thus reducing access and tuning times.
2. Various experiments with different settings verify the effectiveness of the proposed spatial air index method, which outperforms the existing methods *SSI* and *MLAIN*.

This paper extends on previous work reported in [20], providing a more detailed discussion of related work. The proposed spatial air index method is reorganized, and we provide additional comprehensive experiments and discussions.

The rest of this paper is organized as follows. In Section 2, we review related work. Section 3 presents our proposed spatial air index method. Section 4 discusses the experiments. Finally, we conclude this paper in Section 5.

2. Related Work

Recently, many air index methods of organizing spatial data have been designed and implemented to solve energy efficiency and query processing issues in wireless data broadcast. Wireless data broadcast methods can be classified into flat and non-flat approaches based on the broadcast frequency of all data items. In the flat data broadcast approach, all data items are broadcast at the same frequency. In the non-flat approach, hot data items are broadcast more frequently.

2.1. Flat Data Broadcast

Wireless data broadcast systems can be classified as flat and non-flat based on whether clients have a preference for certain data items or not [9,17,21–23]. A flat data broadcast system does not consider clients' preferences for data items: the server broadcasts all data items only once in a broadcast cycle, making it inefficient when clients access certain data items more frequently than others.

In terms of flat data broadcast approaches, Zheng et al. (2004) used the Hilbert curve index (*HCI*) to improve the processing efficiency of window queries and *kNN* queries [24]. The *HCI* was implemented based on B+ tree data structure and interleaved with data items in a (1,*m*) index allocation scheme. Although *HCI* can simulate and approximate the clients' linear access, it consumes significant amounts of energy in computing the queried items. Zheng et al. (2009) later integrated *HCI* and the Distributed Spatial Index (*DSI*) to address this issue [22]. These two schemes used Hilbert Curve (*HC*) values to indicate the location of data items to replace the real coordinates of the data items. *DSI* in a table type is allocated in a distributed scheme. Although this hybrid flat data broadcast approach can expand the queried items and improve query processing by listening sufficiently large candidate items, it consumes significant amounts of energy while computing *HC* values for data items instead of real coordinates.

Lee and Zheng (2005) also used *DSI* and *HC* values to handle query processing in a flat data broadcast system [25]. They designed a distributed table and recorded the data items for quick querying. Although this method can achieve better performance than the traditional *HCI* method, it does not decrease client-side energy consumption. Im et al. [26] extended Lee and Zheng's research [25], presenting a new cell-based distributed index (*CEDI*) method using the two level tables to record the grid space for window queries. To improve access time, the two level tables can allocate a data scheme using an $n \times n$ grid partition. To process given queries, *CEDI* uses real coordinates of data items to outperform *HCI* and *DSI*.

The main drawback of these approaches is that they cannot effectively decrease access time, especially for the non-uniform distribution of data items, which results in significant client-side energy consumption.

2.2. Non-Flat Data Broadcast

Non-flat broadcast systems consider client preferences for data items, providing clients with faster access to hot data items [21]. In a single broadcast cycle, the server prioritizes hot data items for more frequent dissemination, rather than using a uniform frequency for all data items. Clients inform the server of their data item preferences through a low-bandwidth uplink channel [21]. The server responds by broadcasting the requested items through a high-bandwidth downlink channel in a non-flat manner [21]. This decreases the access time for query processing [9,21].

Im et al. (2011) proposed a grid-based distributed index (*GDIN*) [21] for window queries over non-flat broadcast systems using a regular space partition. In this research, the grid cells contain regular cells and hot cells. This method can be easily implemented because the hot cells are used to carry hot data items and regular cells are only used to index the regular data items. However, the access time of *GDIN* cannot be improved because it does not allow for different dissemination frequencies for hot and regular cells.

Im and Choi (2012) later presented a multi-leveled air index scheme (*MLAIN*) [17] to solve the drawback of *GDIN* by using a multi-level space partition. Recently, Im and Hwang (2014) extended their research and presented a two-tier spatial index (*TTSI*) [27] to distinguish between hot and regular data. Although *MLAIN* and *TTSI* handle different data types well, these methods did not allow for hot data to be served at different frequencies. Shen and Jian (in press) proposed a skewed spatial index (*SSI*) [18] to address this problem by classifying popular data into multiple groups, and repeating these groups in the broadcast cycle according to their broadcast frequencies.

The main drawback of non-flat broadcast methods is their high computational cost due to the high degree of overlap between the cells for the broadcast and the query region. The query is executed

even though some of the cells do not contain the desired data items, creating an access time and tuning time bottleneck, which can be treated as a trade-off in the non-flat broadcast methods. To address this issue, this paper proposes a spatial air index method based on the largest empty rectangles.

3. Proposed Spatial Air Index Method

To avoid the unnecessary examination of broadcast content and to accelerate query processing for window queries, we propose a spatial air index method based on largest empty rectangles, named *LER*. We first present the wireless system model used in the proposed method. We then describe the mapping of spatial data from a two-dimensional space to a one-dimensional one. We then show how to generate a non-flat broadcast program in a broadcast cycle and describe how to construct the index structure and largest empty rectangles and insert them into the broadcast program. Finally, we present how mobile devices access the broadcast channel to answer window queries.

3.1. System Model

Figure 1 shows the system model used in our proposed spatial air index method. The server collects mobile clients' preferences from the low-bandwidth uplink channel over a certain period of time. The server updates the access probabilities of the spatial data, and determines their broadcast frequencies. The server then generates a broadcast program in a broadcast cycle into which the corresponding spatial indexes are inserted. Afterward, the server disseminates the broadcast program via the high-bandwidth downlink channel, and repeats the same broadcast program cycle by cycle. Note that the server will update the broadcast program on the next broadcast cycle instead of the current broadcast cycle, if needed. On the other hand, if window queries are issued by mobile clients, the mobile devices tune into the high-bandwidth downlink channel to retrieve the related spatial index and data. The clients may send preference feedback about the spatial data to the server via the low-bandwidth uplink channel.

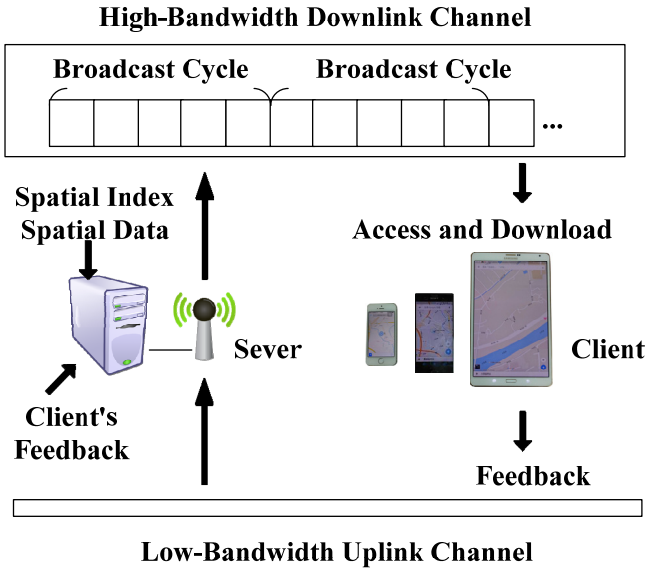

Figure 1. The system model used in the proposed method.

3.2. Mapping of Spatial Data from 2D to 1D

Since access to the wireless channel is linear, in the first step of the proposed method, spatial data in a two-dimensional space should be transformed into a linear order in a one-dimensional space, using a space-filling curve to divide a 2D space into cells, providing each cell with a sequence number, and visiting each cell once in a particular order. Among existing space-filling curves, the Hilbert

Curve (*HC*) is well known for maintaining the spatial locality of the spatial data for the transformation, ensuring that the relative proximity between spatial data items is preserved through the transformation. Therefore, our proposed method uses *HC* to map the spatial data from 2D to 1D.

In the proposed method, we recursively partition a space into four equal-sized cells according to the *HC* until each cell at level n contains at most η spatial objects. That is, the space is partitioned into 2^n by 2^n cells. For each partition at level i, $1 \leq i \leq n$, a visited sequence number called an *HC* value is assigned to each cell. At level i, the *HC* value ranges from 0 to $(2^{i\times 2} - 1)$. In our proposed method, each cell at level i is marked with $C(i,j)$, where j is the *HC* value at level i. We use an example shown in Figure 2a to illustrate the space partition. In Figure 2a, there are 13 restaurants of interest, $\{d_1, \ldots, d_{13}\}$, which are classified into three types: hottest, hot, and normal. Spatial objects d_2 and d_3 are the hottest ones, spatial objects d_8 and d_{12} are hot ones, and the others are normal ones. In this case, with $\eta = 2$, following spatial partitioning, each cell should contain at most two objects. According to the *HC*, at Level 1, the space is partitioned into four equal-sized cells, marked with $C(1,0)$, $C(1,1)$, $C(1,2)$, and $C(1,3)$, as shown in Figure 2b, where the visiting sequence of cells in the *HC* order follows a dotted line with an arrow. Since each cell at Level 1 contains more than two objects, each cell is further partitioned into four equal-sized cells at Level 2, as shown in Figure 2c. At Level 2, to continuously visit the cells, the visited order of the sub-cells in cells $C(1,0)$ and $C(1,3)$ of Level 1 is changed by appropriately following the rotation/flip of that of the cell of Level 1. Following spatial partitioning at Level 2, each cell contains at most two objects, and no further partitioning is needed. In Figure 2c, the cells containing hot objects are named hot cells, which are marked with $H(i,j)$, e.g., $H(2,2)$.

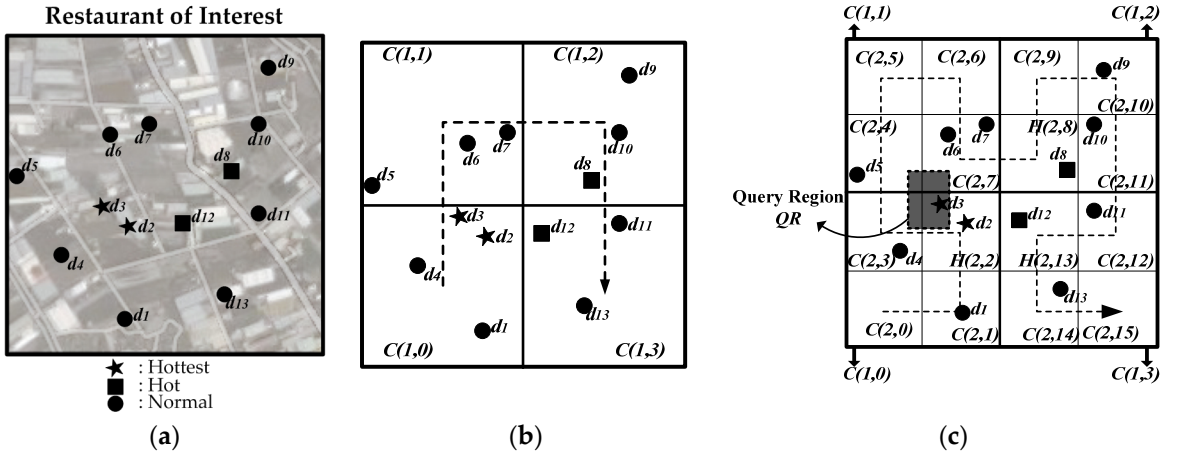

Figure 2. Spatial partitioning according to *HC*: (**a**) the original map; (**b**) spatial partitioning at Level 1; (**c**) spatial partitioning at Level 2.

3.3. Generation of the Non-Flat Broadcast Program

In the second step, based on the skewed access probabilities of the spatial data, the proposed method generates a non-flat broadcast program in a broadcast cycle. Acharya et al.'s broadcast disks (*BD*) method [9] efficiently partitions spatial data into groups (disks) and generates a non-flat broadcast program based on their skewed access patterns, thus achieving quick access for hot spatial data. Therefore, our proposed method adopts the *BD* method to generate the non-flat program. The generation of a broadcast program in a broadcast cycle is processed as follows [9]. First, cells at level n are grouped into S disks according to their access probabilities such that the fast disk repeating more times has few cells and the slow ones repeating fewer times have many cells. Each disk k has a relative frequency λ_k, $1 \leq k \leq S$, which is the number of times it appears in a broadcast cycle. In each disk, the cells have similar access probabilities and are allocated to the *HC* order. Secondly, the least common multiple, *LCM*, of the relative frequencies is calculated to determine how many chunks each disk is split into. Each disk k is further split into (LCM/λ_k) chunks. Finally, one chunk from each disk

is individually scheduled in a round-robin manner to form a minor cycle, B_l, $1 \leq l \leq LCM$, until the last chunk from the last disk is scheduled. That is, there are LCM minor cycles in a broadcast cycle.

Assume $S = 3$ disks and D_1, D_2, and D_3 with $\lambda_1 = 4$, $\lambda_2 = 2$ and $\lambda_3 = 1$, respectively. Take the spatial data in Figure 2c as an example. These cells are grouped into three disks as shown in Figure 3a, where the hot cells appear relatively more frequently. In this case, since $LCM = 4$, disks D_1, D_2, and D_3 are further split into 4/4 (= 1), 4/2 (= 2), and 4/1 (= 4) chunks, respectively, as shown in Figure 3b. Afterward, each chunk from each disk is individually scheduled in circular order to form a non-flat broadcast program until the last chunk from disk D_3 is scheduled, as shown in Figure 3c. In this case, there are four minor cycles in a broadcast cycle, B_1, B_2, B_3, and B_4.

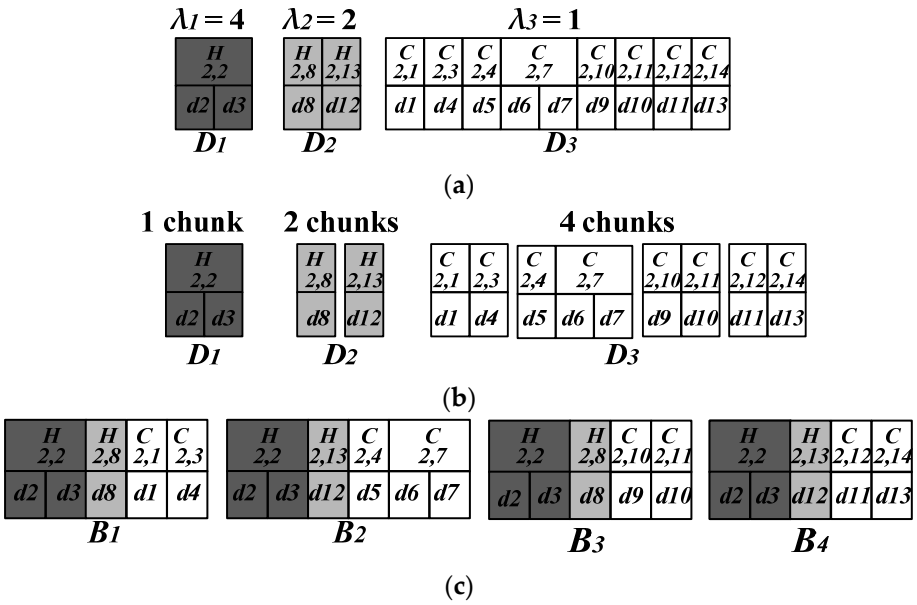

Figure 3. Applying *BD* to generate a non-flat broadcast program: (**a**) grouping cells into disks; (**b**) splitting disks into chunks; (**c**) interleaving chunks to form a broadcast cycle.

3.4. Construction of the Index Structure and Largest Empty Rectangles

Consider that a mobile client has issued a window query with query region QR in Figure 2c. In the previous cell-based methods such as *MLAIN* [17] and *SSI* [18], all the cells that overlap with the query region should be examined. In Figure 2c, although cells $H(2,2)$, $C(2,3)$, $C(2,4)$, and $C(2,7)$ overlap with QR, the only relevant object for this query region is object d_3, which is in $H(2,2)$. In this case, there is no need to examine cells $C(2,3)$, $C(2,4)$, and $C(2,7)$, which increases the access and tuning times. To overcome this problem, we use largest empty rectangles [19] to avoid the unnecessary examination of the broadcast content. A largest empty rectangle is the largest rectangle that does not touch any object. Take cell $H(2,2)$ in Figure 4 as an example to illustrate the largest empty rectangle. Note that the cells in Figure 4 are a portion of the cells in Figure 2c. In Figure 4, the minimum bounding rectangle, MBR_1, is produced to contain objects d_2 and d_3 in cell $H(2,2)$. Afterward, the rectangle of maximal size extending from the edges of MBR_1 without touching any other object forms the largest empty rectangle of cell $H(2,2)$. In Figure 4, since query region QR is totally contained in this largest empty rectangle, the relevant object for this window query is bounded in this cell, and no other cells should be examined. Consequently, the access and tuning times can be reduced. However, if all the largest empty rectangles are inserted in the index structure, the size of a broadcast will become excessively large, leading to a massive increase in access time. Since hot cells are more frequently retrieved than normal ones, only the largest empty rectangles of the hot cells are provided in the index structure of the proposed method.

In the proposed method, three types of index information are provided in the spatial air index structure: the global index, the hot cell index, and the cell index. These indexes are interleaved with the non-flat broadcast program in a broadcast cycle. The insertion of the three index types among the broadcast program proceeds as follows. The global index is inserted before each minor cycle, and the hot cell index before each hot cell. Moreover, the proposed method inserts the cell indexes from Level 1 to $(n-1)$ before the first child cell of the next level that is a non-hot cell. Furthermore, the corresponding cell index of level n is inserted before each cell of level n. Figure 5 shows the non-flat broadcast program for the spatial objects shown in Figure 2c with the distribution of the spatial air index. In Figure 5, four global indexes, GI_1, GI_2, GI_3, and GI_4 are inserted before four minor cycles, and the corresponding hot cell indexes are inserted before each hot cell. The cell index of Level 1 is then inserted before its corresponding first child cell of Level 2 that contains objects and is a non-hot cell. For example, in Figure 5, since cell $C(2,4)$ containing object d_5 is the first child cell inside cell $C(1,1)$, the cell index for $C(1,1)$ is inserted before cell $C(2,4)$. Finally, the corresponding cell index of Level 2 is inserted before each cell of Level 2.

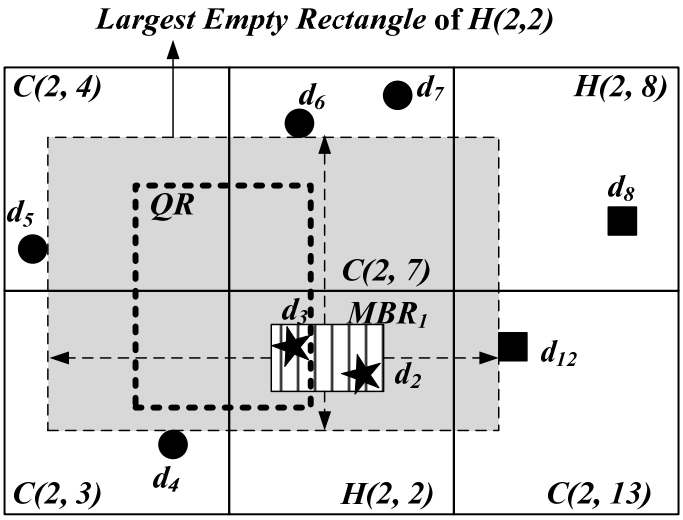

Figure 4. Largest empty rectangle of cell $H(2,2)$.

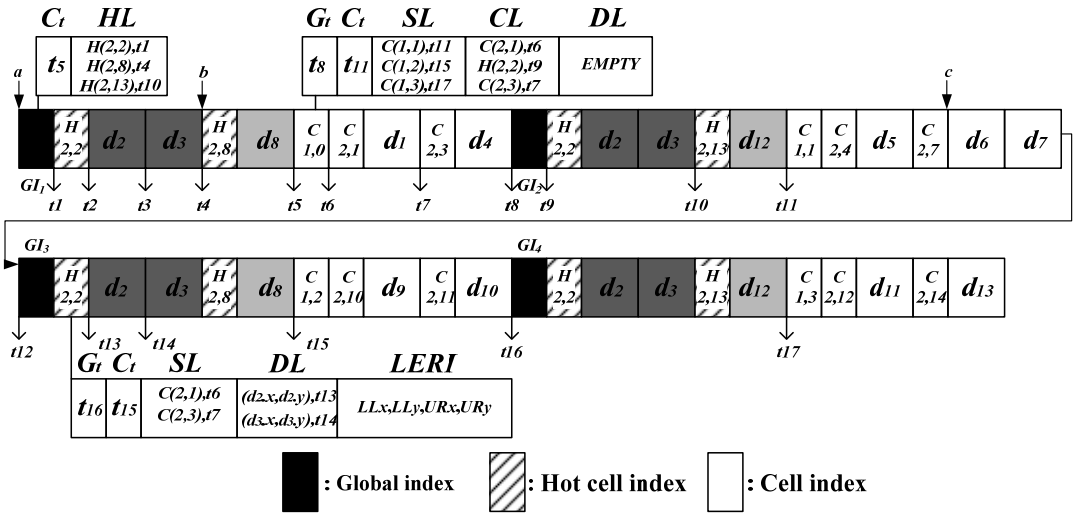

Figure 5. Non-flat broadcast program with the distribution of the spatial air index.

The proposed index structures are as follows.

1. Global index: G = <C_t, HL>

   - C_t is the arrival time of the nearest upcoming cell index of Level 1 in the wireless channel. For example, C_t in the first global index in Figure 5 points to the beginning of cell index C(1,0), t5.
   - HL contains the arrival time of the cell indexes for all hot cells. For example, HL in the first global index in Figure 5 contains information about all the hot cells, H(2,2), H(2,8) and H(2,13).

2. Hot cell index: $H(i,j)$ = <G_t, C_t, SL, DL, $LERI$>

   - G_t is the arrival time of the nearest upcoming global index. For example, in Figure 5, G_t in the hot cell index of H(2,2) in the second row points to the beginning of the upcoming global index, t16.
   - SL contains information about the sibling cells of the same level.
   - DL contains the coordinates of the spatial objects in that cell, and their corresponding arrival times. From DL, the mobile device can determine whether the spatial objects are contained within the query region before retrieving them from the channel.
   - $LERI$ represents the information about the largest empty rectangle of this hot cell. The lower-left and upper-right coordinates of the largest empty rectangle are recorded in $LERI$.

3. Cell index: $C(i,j)$ = <G_t, C_t, SL, CL, DL>

   - CL contains information about the corresponding child cells of the next level. For example, in Figure 5, CL in the cell index of C(1,0) of Level 1 contains information about the child cells of Level 2, C(2,1), H(2,2), and C(2,3).

3.5. Access Processing for Window Queries

To answer a spatial window query, the cells of level n overlapping with the query region should be examined to filter out the relevant spatial objects. Using the largest empty rectangles of the hot cells, the query may quickly terminate early, thus avoiding unnecessary examination of the other overlapping cells. The information about overlapping cells is inserted into the examination set, ES. After receiving a query, the mobile device first tunes in to the wireless channel to obtain the arrival time of the nearest upcoming global index, G_t, and switch into doze mode to wait for the global index. Note that the arrival time of the nearest upcoming global index is also provided in the data segment on the broadcast program, which is not shown in Figure 5. The processing procedures are then handled by Algorithm 1.

Take the window query with query region QR in Figure 2c for example. Cells H(2,2), C(2,3), C(2,4), and C(2,7) overlapping with QR should be examined, i.e., ES = {H(2,2), C(2,3), C(2,4), C(2,7)}. The mobile device tunes in on the wireless channel at point a in Figure 5 and retrieves this global index. After examining HL in this global index, the mobile device has EQ = [t1] and ES = {C(2,3), C(2,4), C(2,7)}. Since ES is not empty, C_t is put into EQ, i.e., EQ = [t1, t5]. The mobile device then retrieves hot cell index H(2,2) at t1. After examining H(2,2), the mobile device verifies that query region QR is contained in $LERI$ of H(2,2), depicted in Figure 4, and the query processing can be finished after retrieving the relevant object in this hot cell. The mobile device then checks DL in H(2,2) with QR, and decides that only spatial object d_3 should be retrieved, i.e., EQ = [t3]. Afterward, the mobile device switches into doze mode and wakes up at t3 to retrieve spatial object d_3, finishing the query processing at point b in Figure 5. In this case, if information about the largest empty rectangle is not provided in the spatial air index, the query process will end at point c in Figure 5. Obviously, the proposed method benefits from using the largest empty rectangle to minimize the access time. Moreover, some cell indexes might be skipped to reduce the tuning time.

Algorithm 1. Process a Window Query

Input: Examination set *ES*.
Output: The relevant objects.

1. Examine *HL* in the global index to obtain the arrival times of the hot cells in *ES*, put those arrival times into examination queue *EQ*, and remove those cells from *ES*. Note that *EQ* is a sorted queue recording the arrival time of the next visited index or object in ascending order.
    a. If *ES* is empty, follow the arrival time in *EQ* to retrieve the cell indexes to obtain the corresponding objects. That is, the query region only overlaps with the hot cells. Otherwise, put the arrival time in C_t into *EQ*.
2. Follow the arrival time in *EQ* to retrieve the cell indexes to obtain the corresponding objects until *ES* is empty. Note that the visited element in *EQ* is removed after it is visited.
    a. If the visited index is the hot cell index, determine whether the query region is contained in *LERI*. If so, the relevant objects are restricted to this hot cell. Follow *DL* in this hot cell index to retrieve the relevant objects and end the query processing. Note that only spatial objects with the coordinates provided in *DL* contained in the query region are retrieved. Otherwise, examine *SL* in this hot cell with *ES*, put the corresponding arrival time into *EQ*, and decide whether C_t should be put into *EQ*.
    b. If the visited index is the cell index, examine *SL*, *CL*, and *DL* in this cell index with *ES*, put the corresponding arrival time into *EQ*, and decide whether C_t should be put into *EQ*. Follow *DL* in this cell index to retrieve the relevant objects if necessary.

4. Simulation Study

The experimental results shown in [17] verified that *MLAIN* outperforms *HCI* [24], *DSI* [22], and *GDIN* [21]. Moreover, results shown in [18] verified that *SSI* outperforms *MLAIN*. Therefore, in the performance evaluation, we choose these two spatial air index methods, *MLAIN* [17] and *SSI* [18], for non-flat wireless broadcast to compare with our proposed method, *LER*. The average access time, tuning time, and energy consumption are of concern in the experimental results.

4.1. Simulation Model

In our simulation study, to evaluate the performance among the difference methods, we use a real data set as shown in Figure 6, which contains 5922 points of interest in Greece. Buckets are logical transmission units in a wireless broadcast channel, and spatial indexes and objects are allocated to the buckets. In our simulation, we assume that a bucket occupies 64 bytes, and a spatial object occupies 1024 bytes. Moreover, the coordinates of a spatial object are represented by two one-byte integers.

Figure 6. Real dataset from Greece.

Consider that the dataset contains N spatial objects. To simulate the skewed access patterns of spatial objects for mobile clients, we assign the access probability, $Pr(i)$, $1 \leq i \leq N$, of each spatial object i by Equation (1), which follows the *Zipf* distribution [9]:

$$Pr\,(i) = \frac{(1/i)^{\theta}}{\sum_{j=1}^{N}(1/j)^{\theta}}, \tag{1}$$

where θ is the *Zipf* factor to control the skewness of the distribution. With the increase in the value of θ, the distribution of access probabilities of spatial objects will be highly skewed. Given the skewed access patterns of spatial objects generated by the *Zipf* distribution in Equation (1), we apply Yee et al.'s *GREEDY* algorithm [28] to partition these spatial objects into S disks, which can minimize the average access time of the non-flat broadcast program generated by the broadcast disks method [9]. In our experiment, we consider that any spatial object on the same disk to be uniformly accessed. Therefore, we also assign the demand access probability, $D\_Pr(k)$, for each disk k, $1 \leq k \leq S$, according to the *Zipf* distribution with *Zipf* factor ϕ by Equation (2). That is, the skewness of issuing window queries is applied among the disks. $D\_Pr(k)$ affects the ratio of queries issued among the disks. With the increase on the value of ϕ, there are more queries issued on the fast disk and relatively few issued on the slow disks.

$$D\_Pr\,(k) = \frac{(1/k)^{\varphi}}{\sum_{j=1}^{S}(1/j)^{\varphi}} \tag{2}$$

Moreover, we assign the relative frequency, R_k, of each disk k, $1 \leq k \leq S$, by Equation (3) [9].

$$R_k / R_S = (S - k)\,\Delta + 1, \tag{3}$$

where $R_S = 1$ and Δ is the factor to control relative frequencies among disks. In our experimental results, the number of disks, S, is set to 3, and the value of the factor for relative frequencies, Δ, is set to 1.

In our simulation study, we use the parameter *WinLengthRatio* to control the search region of a window query. *WinLengthRatio* is the ratio of the side length of the query region to that of a map. Assuming that the side length of a map is SL, the query region is a square with a side length equal to (SL * *WinLengthRatio*). Moreover, since our *LER* method is proposed to avoid the unnecessary examination of cells adjacent to the designated cell, we use the parameter *Crosspro* to control the probability that a query region crosses more than one cell. For example, if *Crosspro* is set to 60%, the query region has a 60% probability of spanning two cells. Table 1 summarizes the parameters used in our experiments.

For each experimental result, 10,000 window queries are issued among the real dataset, and the performance results are the average among these queries. The average access and tuning times are measured in terms of buckets [15]. In [17,18], to evaluate the energy consumption of mobile devices, the energy consumption of their CPU and network interface card is considered. The CPU consumes 400 mW/s and 0.16 mW/s in the active and doze modes, respectively. The network interface card consumes 750 mW/s and 25 mW/s in the active and doze modes, respectively. Therefore, a mobile device consumes 1150 mW/s and 25.16 mW/s in the active and doze modes, respectively. In our simulation, we use the same assumption of the energy consumption of a mobile device, as used in [17,18]. Moreover, we assume a downlink channel bandwidth of 1 Mbps. As a result, the average energy consumption, *Avg_E*, in the unit of the joule (J) for processing window queries can be determined by Equation (4), where *Avg_AT* and *Avg_TT* are the average access and tuning times, respectively, in seconds.

$$Avg\_E = 1150 \times Avg\_TT + 25.16 \times (Avg\_AT - Avg\_TT) \tag{4}$$

Table 1. Performance parameters and their default values.

Parameter	Default Value	Meaning
θ	2.5	The *Zipf* factor to control the access skewness of the distribution of spatial objects
ϕ	2	The *Zipf* factor to control the access skewness of the distribution of disks
WinLengthRatio	0.11%	The ratio of the side length of the query region to that of a map
Crosspro	60%	The probability that a query region is crossing more than one cell

4.2. Simulation Results

In our experimental results, we evaluate the impact of the performance parameters *Crosspro*, *WinLengthRatio*, θ and ϕ. By changing the value of each parameter above, the other parameters are set to the default values as shown in Table 1.

4.2.1. Impact of *Crosspro*

In the first experimental result, we evaluate the impact of *Crosspro* with values of 20%, 40%, 60%, 80%, and 100%. Figure 7 shows the average access time. The average access times of the three methods increase as the value of *Crosspro* increases from 20% to 100%. As the value of *Crosspro* increases, the probability that a query region will cross more than one cell increases, thus increasing the likelihood that adjacent cells will be examined. As a result, the average access time increases with the value of *Crosspro*. However, Figure 7 shows that the average access time of *MLAIN* and *SSI* increases dramatically with the value of *Crosspro*, whereas the value of *LER* increases smoothly due to the use of largest empty rectangles. If the query region is restricted to the largest empty rectangle, *LER* will not process the other cells, thus reducing the average access time by 56.4% and 24.2%, respectively, in contrast to *MLAIN* and *SSI*.

Table 2 lists the corresponding average tuning time. As the value of *Crosspro* increases, the probability that the query region will overlap with more than one cell increases. In such cases, more relevant objects in the query region will be retrieved, increasing the average tuning time. This trend can be observed in Table 2 for the three methods. Moreover, since our proposed *LER* adopts largest empty rectangles in the spatial index, the query processing ends if the query region is limited within one. Therefore, in Table 2, as the value of *Crosspro* increases, the percentage of the average tuning time of *LER* to both *MLAIN* and *SSI* reduces. Note that in Table 2, the reduction percentage of the average tuning time of *LER* in contrast to the other methods is shown in parentheses. On average, *LER* produces a respective reduction of average tuning time of 8.6% and 8.2% against *MLAIN* and *SSI*. Moreover, Figure 8 shows the corresponding average energy consumption. On average, *LER* produces a 55.8% and 23.2% respective reduction in average energy consumption in contrast to *MLAIN* and *SSI*.

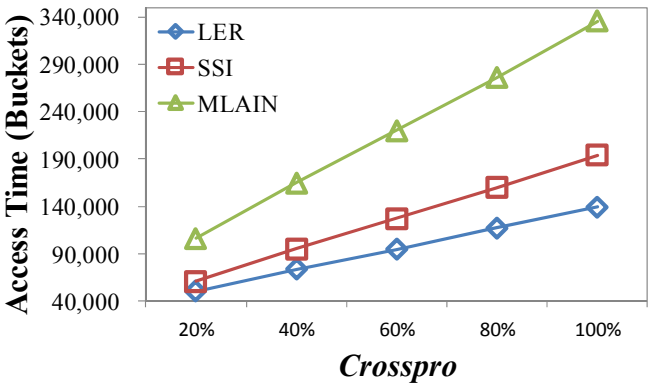

Figure 7. Average access time for changing *Crosspro* values.

Table 2. Average tuning time for changing *Crosspro* values.

Crosspro	LER	SSI	MLAIN
20%	83.63	87.56 (4%)	87.72 (5%)
40%	123.56	133.51 (7%)	133.76 (8%)
60%	160.70	176.85 (9%)	177.19 (9%)
80%	201.04	222.46 (10%)	222.91 (10%)
100%	238.93	267.23 (11%)	267.76 (11%)

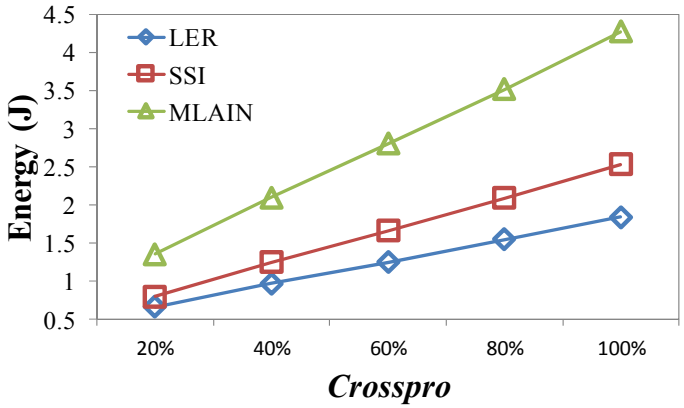

Figure 8. Average energy consumption for changing *Crosspro* values.

4.2.2. Impact of *WinLengthRatio*

We evaluated the impact of *WinLengthRatio* with values of 0.08%, 0.095%, 0.11%, 0.125%, and 0.14%. Figure 9 shows the average access time. Among the three methods, *LER* has the shortest average access time. Moreover, on average, *LER* produces a respective 57% and 25.4% reduction on the average access time over *MLAIN* and *SSI*. The three methods show no significant increase for average access times as the value of *WinLengthRatio* increases. This observation can be explained as follows. The span of the relevant objects in a query region has a profound influence on access times. Note that the span refers to the distance between the first relevant object and the final one in the wireless channel. In the experimental result, the spans of the relevant objects for the query regions with different values of *WinLengthRatio* are quite similar. Therefore, the average access times for each method increased slightly as the values of *WinLengthRatio* increased.

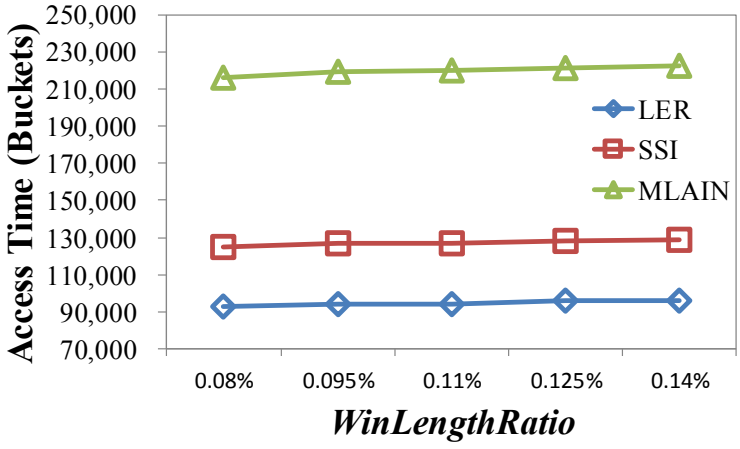

Figure 9. Average access time for changing *WinLengthRatio* values.

Table 3 lists the corresponding average tuning time. The average tuning times for the three methods increase with the value of *WinLengthRatio*. As the value of *WinLengthRatio* increases, the query region increases and covers more the relevant objects. Therefore, more spatial indexes and the relevant spatial objects should be examined, resulting in an increase to the average tuning time. However, due to the benefit from inserting information about largest empty rectangles in the index structure, *LER* can reduce the tuning time as the same case in both *MLAIN* and *SSI*. In Table 3, *LER* has the shortest average tuning time among the three methods. On average, *LER* produces a respective 9.8% and 9.6% reduction to the average tuning time compared to *MLAIN* and *SSI*. Moreover, Figure 10 shows the corresponding average energy consumption; on average, our proposed method produces respective 55.2% and 24.4% reductions to the average energy consumption over *MLAIN* and *SSI*.

Table 3. Average tuning time for changing *WinLengthRatio* values.

WinLengthRatio	LER	SSI	MLAIN
0.08%	77.20	87.64 (12%)	87.77 (12%)
0.095%	117.68	130.36 (10%)	130.52 (10%)
0.11%	161.68	177.67 (9%)	178.03 (9%)
0.125%	224.40	245.50 (9%)	246.95 (9%)
0.14%	310.58	337.48 (8%)	339.58 (9%)

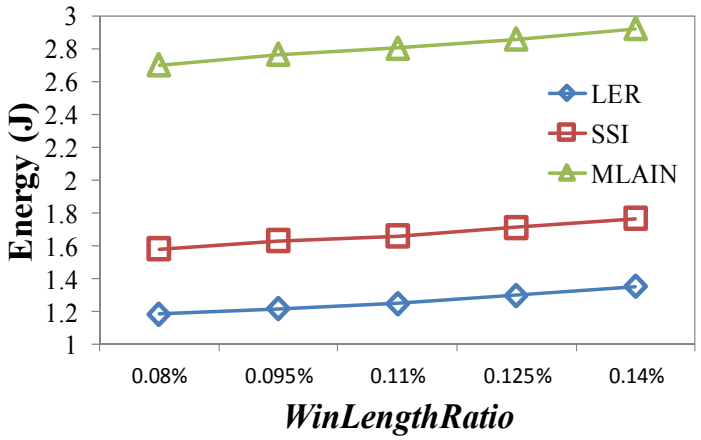

Figure 10. Average energy consumption for changing *WinLengthRatio* values.

4.2.3. Impact of θ

In the third experimental result, we evaluate the impact of θ with values of 2.1, 2.3, 2.5, 2.7, and 2.9. Figure 11 shows the average access time. As the value of θ increases, the access probabilities of spatial objects become skewed. That is, a few spatial objects in the hot cells with high access probability appear more frequently in the broadcast cycle. This leads to a reduction in the average access time. In Figure 11, as the value of θ increases, the average access times of the three methods decrease slightly, except for those of *MLAIN* and *SSI*, for which the value of θ increases from 2.1 to 2.3. Moreover, the access times of *MLAIN* are much longer than those of *LER* and *SSI*. In *MLAIN*, spatial objects are partitioned into two groups: hot and normal. All the spatial objects in the hot group appear many times at a relatively high broadcast frequency in a broadcast cycle. Therefore, the number of spatial objects in this hot group has a strong effect on the size of the broadcast cycle, which determines the access time. On the other hand, in both *LER* and *SSI*, the spatial objects are partitioned into a few groups by the *BD* method [9], instead of just two groups. Consequently, the broadcast cycle in both *LER* and *SSI* is smaller than that in *MLAIN*. Thus, the access time for *MLAIN* is much longer than that of *LER* and *SSI*. Furthermore, using largest empty rectangles in the spatial air index, *LER* has a shorter access time than *SSI*. Our experimental results show that, on average, *LER* reduces the average access

time by 57.4% and 27.6%, respectively, compared to *MLAIN* and *SSI*. Table 4 lists the corresponding average tuning time. On average, *LER* produces a respective 9.8% and 9.6% to the average tuning time of *MLAIN* and *SSI*. Moreover, Figure 12 shows that, on average, *LER* produces a 55.6% and 26.6% respective reduction to average energy consumption for *MLAIN* and *SSI*.

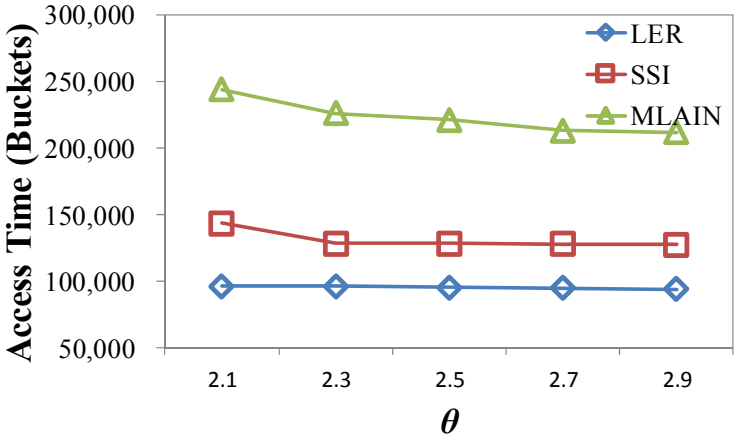

Figure 11. Average access time for changing θ values.

Table 4. Average tuning time for changing of θ values.

θ	LER	SSI	MLAIN
2.1	182.65	201.82 (9%)	202.21 (10%)
2.3	162.77	178.73 (9%)	179.05 (9%)
2.5	162.31	178.12 (9%)	178.47 (9%)
2.7	159.38	177.73 (11%)	178.07 (11%)
2.9	158.88	176.76 (10%)	177.10 (10%)

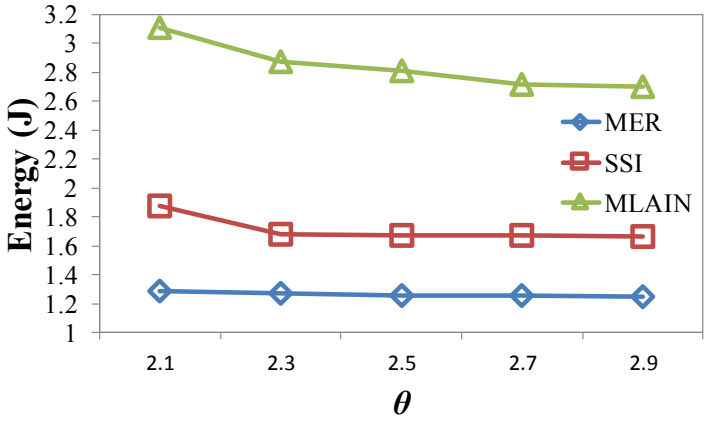

Figure 12. Average energy consumption for changing θ values.

4.2.4. Impact of ϕ

In the final experimental result, we evaluate the impact of ϕ with values of 1, 1.5, 2, 2.5, and 3. Figure 13 shows the average access time. The skewness of window queries among disks increases with the value of ϕ. That is, the number of window queries issued for the cells on the fast disk grows and that for the cells on the slow disk declines. Since the spatial objects on the fast disk appear many times in a broadcast cycle, the access time for retrieving them is shorter than that on the slow disk. As a result, the average access time is reduced as the value of ϕ increases. This trend can be seen in the three methods in Figure 13. *LER* has the shortest average access time among the three methods.

On average, *LER* produces a respective 56.2% and 25.2% reduction to the average access time compared to *MLAIN* and *SSI*. Table 5 lists the corresponding average tuning times. Similar to the average access time, the average tuning time decreases as the value of ϕ increases. Since *LER* uses largest empty rectangles for the hot cells to avoid unnecessary examination of the broadcast content, *LER* can reduce the tuning time for processing window queries. In Table 5, *LER* has the shortest average tuning time among the three methods, 8.8% less than both *MLAIN* and *SSI*. Moreover, Figure 14 shows that the average energy consumption of *LER* is 54.8% and 24.2% less, respectively, than that of *MLAIN* and *SSI*.

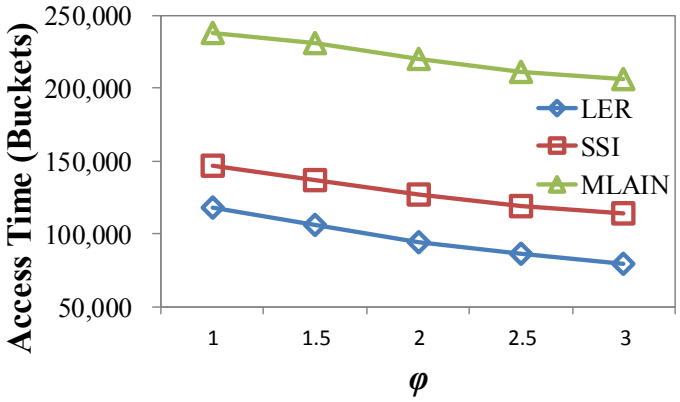

Figure 13. Average access time for changing ϕ values.

Table 5. Average tuning time for changing ϕ values.

ϕ	LER	SSI	MLAIN
1	185.62	199.99 (7%)	200.54 (7%)
1.5	173.56	189.16 (8%)	189.61 (8%)
2	161.77	177.74 (9%)	178.09 (9%)
2.5	150.24	166.36 (10%)	166.61 (10%)
3	142.98	159.07 (10%)	159.25 (10%)

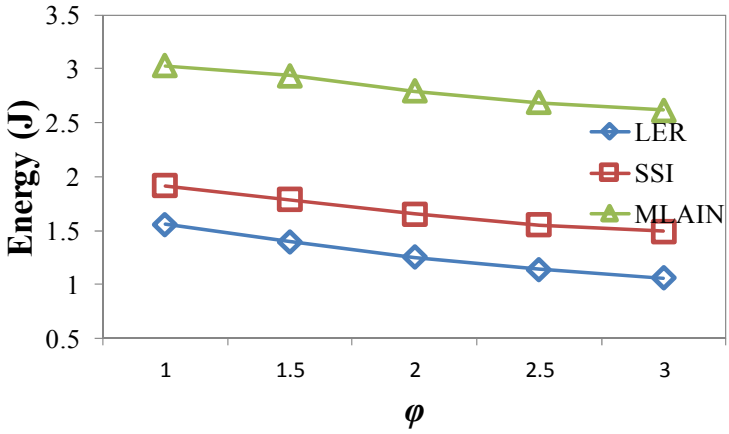

Figure 14. Average energy consumption for changing ϕ values.

5. Conclusions

This paper considers skewed access patterns of mobile clients in proposing a spatial air index method to efficiently handle the access processing of window queries in non-flat wireless broadcast environments. Our proposed spatial air index method uses largest empty rectangles of hot cells to reduce unnecessary access of invalid cells broadcast on the wireless channel. Therefore, while

processing window queries, mobile devices can skip the unnecessary examination of the broadcast content and thus accelerate query processing termination and reducing access time, tuning time, and energy consumption. Performance evaluation shows that our proposed spatial air index method outperforms the existing methods *MLAIN* and *SSI* in terms of average access time, tuning time, and energy consumption, with average respective improvements of 57.4%, 9.8%, and 55.8% over *MLAIN*, and 27.6%, 9.6%, and 26.6% over *SSI*. Future work will investigate how to support different types of spatial queries using the largest maximum rectangle technique.

Acknowledgments: This research was supported by grants MOST 105-2410-H-468-012 and MOST 105-2410-H-025-015-MY2 from the Ministry of Science and Technology, Taiwan.

Author Contributions: Jun-Hong Shen and Ching-Ta Lu provided the core idea for this research; Jun-Hong Shen implemented the methodology of this study; Jun-Hong Shen, Mu-Yen Chen, and Chien-Tang Mai designed and performed the experiments; Jun-Hong Shen and Mu-Yen Chen wrote the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shen, J.H.; Lu, C.T.; Jian, M.S.; Huang, T.C. A skewed spatial index for continuous window queries in the wireless broadcast environments. *Lect. Notes Elect. Eng.* **2013**, *253*, 702–715.
2. Shen, J.H.; Lu, C.T.; Jian, M.S. Neighbor-index method for continuous window queries over wireless. *Appl. Mech. Mater.* **2013**, *284*, 3295–3299. [CrossRef]
3. Dhanked, S.; Goel, V. Hashing techniques for broadcasting in wireless data environment. In Proceedings of the International Conference on Issues and Challenges in Intelligent Computing Techniques, Ghaziabad, India, 7–8 February 2014.
4. Gao, X.; Yang, Y.; Chen, G.; Lu, X. Global optimization for multi-channel wireless data broadcast with AH-tree indexing scheme. *IEEE Trans. Comput.* **2016**, *65*, 2104–2117. [CrossRef]
5. Goel, V.; Ahalawat, A.K.; Gupta, M.N. Distributed air indexing scheme for full-text search on multiple wireless channel. In *Intelligent Systems Technologies and Applications*; Berretti, S., Thampi, S.M., Dasgupta, S., Eds.; Springer: Berlin, Germany, 2016; pp. 125–135.
6. Vishnoi, S.; Goel, V. Novel table based air indexing technique for full text search. In Proceedings of the International Conference on Computational Intelligence and Communication Technology, Ghaziabad, India, 13–14 February 2015.
7. Zhong, J.; Wu, W.; Gao, X.; Shi, Y.; Yue, X. Evaluation and comparison of various indexing schemes in single-channel broadcast communication environment. *Knowl. Inf. Syst.* **2014**, *40*, 375–409. [CrossRef]
8. Rakotonirainy, A.; Obst, P.; Loke, S.W. Socially aware computing constructs. *Int. J. Soc. Humanist. Comput.* **2012**, *1*, 375–395. [CrossRef]
9. Acharya, S.; Franklin, M.; Zdonik, S.; Alongso, R. Broadcast disks: Data management for asymmetric communications environments. In Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, CA, USA, 22–25 May 1995.
10. Shen, J.H.; Chang, Y.I. A skewed distributed indexing for skewed access patterns on the wireless broadcast. *J. Syst. Softw.* **2007**, *80*, 711–723. [CrossRef]
11. Im, S.; Joo, K.H.; Choi, J. Efficient windows query processing with expanded grid cells on wireless spatial data broadcasting for pervasive computing. *Contemp. Eng. Sci.* **2014**, *7*, 785–790. [CrossRef]
12. Park, K.; Valduriez, P. A hierarchical grid index (HGI), spatial queries in wireless data broadcasting. *Distrib. Parallel Databases* **2013**, *31*, 413–446. [CrossRef]
13. Park, K. An efficient scalable spatial data search for location-aware mobile services. *J. Inf. Sci. Eng.* **2015**, *31*, 165–178.
14. Jung, S.; Mo, H. A spatial indexing scheme for location based service queries in a single wireless broadcast channel. *J. Inf. Sci. Eng.* **2014**, *30*, 1945–1963.
15. Shen, J.H.; Chang, Y.I. An efficient nonuniform index in the wireless broadcast environments. *J. Syst. Softw.* **2008**, *81*, 2091–2103. [CrossRef]
16. Song, D.; Park, K. A partial index for distributed broadcasting in wireless mobile networks. *Inf. Sci.* **2016**, *348*, 142–152. [CrossRef]

17. Im, S.; Choi, J. MLAIN: Multi-leveled air indexing scheme in non-flat wireless data broadcast for efficient window query processing. *Comput. Math. Appl.* **2012**, *64*, 1242–1251. [CrossRef]
18. Shen, J.H.; Jian, M.S. Spatial query processing for skewed access patterns in nonuniform wireless data broadcast environments. *Int. J. Ad Hoc Ubiquitous Comput.* **2016**, in press.
19. Aggarwal, A.; Suri, S. Fast algorithms for computing the largest empty rectangle. In Proceedings of the Third Annual Symposium on Computational Geometry, Waterloo, ON, Canada, 8–10 June 1987.
20. Shen, J.H.; Lu, C.T.; Mai, C.T. Efficient processing of spatial window queries for non-flat wireless broadcast. In Proceedings of the 5th International Conference on Frontier Computing, Tokyo, Japan, 13–15 July 2016.
21. Im, S.; Youn, H.Y.; Choi, J.; Ouyang, J. A novel air indexing scheme for window query in non-flat wireless spatial data broadcast. *J. Commun. Net.* **2011**, *13*, 400–407. [CrossRef]
22. Zheng, B.; Lee, W.C.; Lee, C.K.; Lee, D.L.; Shao, M. A distributed spatial index for error-prone wireless data broadcast. *VLDB J.* **2009**, *18*, 959–986. [CrossRef]
23. Vaidya, N.H.; Hameed, S. Scheduling data broadcast in asymmetric communication environments. *Wirel. Net.* **1999**, *5*, 171–182. [CrossRef]
24. Zheng, B.; Lee, W.C.; Lee, D.L. Spatial queries in wireless broadcast systems. *Wirel. Net.* **2004**, *10*, 723–736. [CrossRef]
25. Lee, W.; Zheng, B. DSI: A fully distributed spatial index for location-based wireless broadcast services. In Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS), Columbus, OH, USA, 6–10 June 2005.
26. Im, M.; Song, J.; Kim, S.K.; Hwang, C. An error-resilient cell-based distributed index for location-based wireless broadcast services. In Proceedings of the 5th ACM International Workshop on Data Engineering for Wireless and Mobile Access, Chicago, IL, USA, 25–28 June 2006.
27. Im, S.; Hwang, H. A two-tier spatial index for non-flat spatial data broadcasting on air. *IEICE Trans. Commun.* **2014**, *97*, 2809–2818. [CrossRef]
28. Yee, W.G.; Navathe, S.B.; Omiecinski, E. Efficient data allocation over multiple channels at broadcast servers. *IEEE Trans. Comput.* **2002**, *51*, 1231–1236.