An Efficient Adaptive Bitmap-based Selective Tuning Scheme for Spatial Queries in Broadcast Environments

Doohee Song and Kwangjin Park
School of Electrical Electronics and Information Engineering Wonkwang University
Iksan-shi, Chunrabuk-do 570-749, Republic of Korea
[e-mail: songdoohee@naver.com, kjpark@wku.ac.kr]
*Corresponding author: Kwangjin Park

Received March 29, 2011; revised July 30, 2011; revised October 18, 2011; accepted October 27, 2011; published October 31, 2011

Abstract

With the advances in wireless communication technology and the advent of smartphones, research on location-based services (LBSs) is being actively carried out. In particular, several spatial index methods have been proposed to provide efficient LBSs. However, finding an optimal indexing method that balances query performance and index size remains a challenge in the case of wireless environments that have limited channel bandwidths and device resources (computational power, memory, and battery power). Thus, mechanisms that make existing spatial indexing techniques more efficient and highly applicable in resource-limited environments should be studied.

Bitmap-based Spatial Indexing (BSI) has been designed to support LBSs, especially in wireless broadcast environments. However, the access latency in BSI is extremely large because of the large size of the bitmap, and this may lead to increases in the search time. In this paper, we introduce a Selective Bitmap-based Spatial Indexing (SBSI) technique. Then, we propose an Adaptive Bitmap-based Spatial Indexing (ABSI) to improve the tuning time in the proposed SBSI scheme. The ABSI is applied to the distribution of geographical objects in a grid by using the Hilbert curve (HC). With the information in the ABSI, grid cells that have no objects placed, (i.e., 0-bit information in the spatial bitmap index) are not tuned during a search. This leads to an improvement in the tuning time on the client side. We have carried out a performance evaluation and demonstrated that our SBSI and ABSI techniques outperform the existing bitmap-based DSI (B·DSI) technique.

Keywords: Location-based service, k-nearest neighbors, wireless broadcast, spatial database, query processing

The part of this paper was presented in the ICONI (International Conference on Internet) 2010, December 16-20, 2010, Philippines. This paper was supported by Wonkwang University in 2011.

DOI: 10.3837/tiis.2011.10.011

1. Introduction

In Location-Based Services (LBSs), location information is used along with digital maps, which are obtained from Global Positioning Systems (GPSs) and mobile communications networks, to provide users with useful information [1][2]. In the past, most LBSs were provided using an on-demand method [3][4]; however, recently, many studies have been conducted on the broadcasting method [5][6][7]. The reason for this change is that as the number of users of LBSs gradually increases, so the number of queries increases; however, the server's capability to process queries is limited. Such a limitation can be overcome by using the broadcasting method. LBSs have become closely connected with, and hence, ubiquitous in, our daily lives [8][9][10].

The techniques most frequently used in LBSs include the range query and the *k*-nearest neighbor (*k*NN) query [11][12][13]. A range query is a query processing method that is used to find a desired object within a fixed range, while a *k*NN query is an operation that is used to find the *k* objects that are closest to the query point. In reference [11], the Distributed Spatial Indexing (DSI) technique was proposed, since it can be applied to reduce the size of the index. Subsequently, the Bitmap-based Spatial Indexing (BSI) [14] technique was proposed as a technique facilitating the application and development of the DSI technique. BSI is an effective technique for determining whether an object exists. In this technique, values (Hilbert curve values) corresponding to each object are expressed as bits (1, 0) and are based on the Hilbert curve index (HCI) [15]. However, the BSI technique has the drawback that the size of the index changes depending on the last HC value in the bitmap. Thus, increasing HC value influence directly the size of the index. Further, an increase in the HC value increases the overall broadcast cycle and the access latency, and it decreases the level of user satisfaction because the server's processing capacity remains limited. As mentioned earlier, in previous studies on LBSs, researchers focused on reading data efficiently, whereas techniques for reading indexes or reducing the size of indexes have not been given sufficient attention. To address this topic, this paper proposes the Selective Bitmap-based Spatial Indexing (SBSI) scheme, as well as an Adaptive Bitmap-based Spatial Indexing (ABSI) that is exploited in the proposed SBSI scheme to achieve a reduced tuning time on the client side.

In the proposed SBSI scheme, the existence/absence of objects is verified using the spatial bitmap index information, and the distance of objects from the query point is measured. On the basis of this measured information, selective tune to the relevant bitmap index is possible, which increases the query processing efficiency.

The ABSI reduces the tuning time on the client side by skipping unnecessary search points (i.e., bitmap index entries not relevant to the submitted query are quickly excluded from the search). A square grid (size: $n \times n$) is formed using the HC value and is segmented into cells of value δ (ideal δ value shall be determined by server). Each cell in the grid has a bit value of 1 when a spatial object exists in the cell and a bit value of 0 otherwise. This ABSI is applied to the proposed SBSI scheme to speed up spatial query operations. Our contributions are summarized as follows:

1. We introduce the SBSI scheme to reduce the access latency and tuning time in the B·DSI scheme. The proposed SBSI scheme selectively tunes the bitmap index and data arrival time information on the basis of whether the size of the data sent from the server is identical or varying.
2. We propose an ABSI that can be used to enhance the query performance of the SBSI scheme. The searching algorithms utilize the information in the ABSI to exclude 0-bit

bitmap index data that have no object information from spatial query operations. This leads to reduced tuning time on the client side.

3. We implement our proposed schemes to validate the performance of our algorithms. Experimental results show that the proposed algorithms help to improve performance.

The rest of this paper is organized as follows. In Section 2, some related studies are introduced, while in Section 3, the algorithms for the SBSI and ABSI techniques are proposed. In Section 4, experimental results are presented, and conclusions to this work are provided in Section 5.

2. Related Work

In conventional on-demand services, indexes are organized based on R-trees [16] or X-trees [17]. The most commonly used index structures are R-trees, which have several variants such as R*-trees [18] and R^+-trees [19]. Index structures in the R-tree family are suitable for disk-based databases, but the use of R-trees in wireless environments accompanies backtracking that causes data access delay [20]. A considerable amount of work has been conducted to find solutions to this problem.

2.1 Broadcast Model

Mobile devices equipped with high-speed wireless networks and GPS offer the advantage that users can obtain information relevant to their location at any time and in any location. However, there are drawbacks of wireless broadcasting, which include the narrow network bandwidth and the limitations imposed on the battery capacity of the user's mobile device. In [21][22][23], a solution to the abovementioned problem, i.e., the limitation of the battery capacity of the user's mobile device, was suggested. In [21][22][23], the battery usage of the mobile device was reduced by means of selective tuning. The terms related to wireless indexing are defined below.

- Probe wait: the time required to begin indexing after the user submits a query.
- Access latency: the total amount of time elapsed between the time at which the user submits a query and the time at which the desired data are delivered.
- Tuning time: the total amount of time for which the client is in active mode in order for the requested data to be delivered.

2.2 Distributed Spatial Indexing

A Distributed Spatial Indexing (DSI) proposes technique in which the distributed scheme index [24] is used and spatial information is considered [11]. In a DSI technique based on Hilbert-type space-filling curves [25], the distributed index is inserted between each data object (shaded rectangles in **Fig. 1**) instead of among all the data information from {0} through {5}, as shown in **Fig. 1**. The distributed index represents information on the location of an object, with increments of size 2^n. This means that the index of {1} is used to search {1, 2, 3, 5}. However, a major disadvantage of the DSI technique is that an increasing amount of tuning time is spent to check {4}, which is not verified because the index increases exponentially by a size of 2^n.

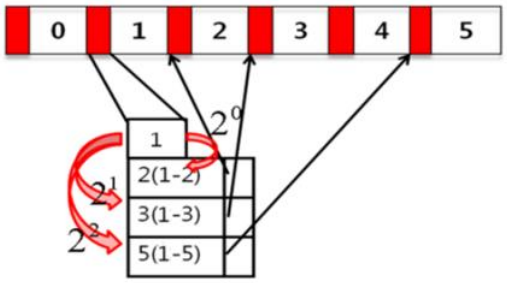


Fig. 1. DSI diagram

2.3 Bitmap-based Spatial Indexing

The Bitmap-based Spatial Indexing (BSI) scheme [14] used bitmap index [26][27]. The BSI supports LBSs in wireless broadcasting environments and involves the use of bits (0, 1) to indicate whether or not data exist for each HC value, as shown in **Fig. 2**. If the HC value has data, the bit is 1; if not, the bit is 0. We can explain this by taking the example shown in **Fig. 2**: in the map of HC order 3, there are 64 spaces from No. 0 to 63. In these 64 spaces, there are objects in Nos. 3, 7, 15, 35, 54, and 62, all of which are represented as bit 1. The remaining spaces, where there are no objects, are represented as bit 0.

21 0	22 0	25 0	26 0	37 0	38 0	41 0	42 0
20 0	23 0	24 0	27 0	36 0	39 0	40 0	43 0
19 0	18 0	29 0	28 0	35 1•	34 0	45 0	44 0
16 0	17 0	30 0	31 0	32 0	33 0	46 0	47 0
15 1•	12 0	11 0	10 0	53 0	52 0	51 0	48 0
14 0	13 0	8 0	9 0	54 1•	55 0	50 0	49 0
1 0	2 0	7 1•	6 0	57 0	56 0	61 0	62 1•
0 0	3 1•	4 0	5 0	58 0	59 0	60 0	63 0

Bitmap index =>$00010\ldots0010_{(2)}$

Fig. 2. BSI diagram

In this paper, authors propose a bitmap-based DSI (B·DSI) technique. The advantage of using the bitmap is that one bit can be used to check whether or not an object exists. Through 64 bits, the data on the objects of **Fig. 2** can be comprehended, and additionally, the distance from the query point can be measured. However, a disadvantage of using the bitmap is that the size of the index increases as the HC value increases. In particular, a possible problem is that if the amount of data is small and the HC value increases, the bitmap index size may become too

large for the given amount of data.

2.4 Distributed Bitmap-based Distributed Spatial Indexing

In the proposed Distributed Bitmap-based Distribted Spatial Indexing (DB·DSI) technique [28], as a solution to the problems faced in B·DSI, first, the whole broadcast cycle is reduced by removing unnecessary procedures such as reading unverified data, in accordance with the 2^n exponential increase in the DSI index. In BSI, the increase in the number of the HC order N (4^n) leads directly to an increase in the index size. As a solution to this problem, we propose a DB·DSI technique for reducing the index size.

Fig. 3 (a) and (b) show the index structures of B·DSI and DB·BSI on the basis of the representation in Fig. 2. According to the B·DSI index structure in Fig. 3 (a), the size of B·DSI is 64 bits, since all objects are inserted. On the other hand, the DB·DSI index structure in Fig. 3 (b) shows that the size was reduced by 41 bits to 23 bits, owing to the insertion of as much as DSI index (refer to the Fig. 1), which is the amount by which the index size of the DSI increased. The increase in number of HC order is translated into the difference in the index size.

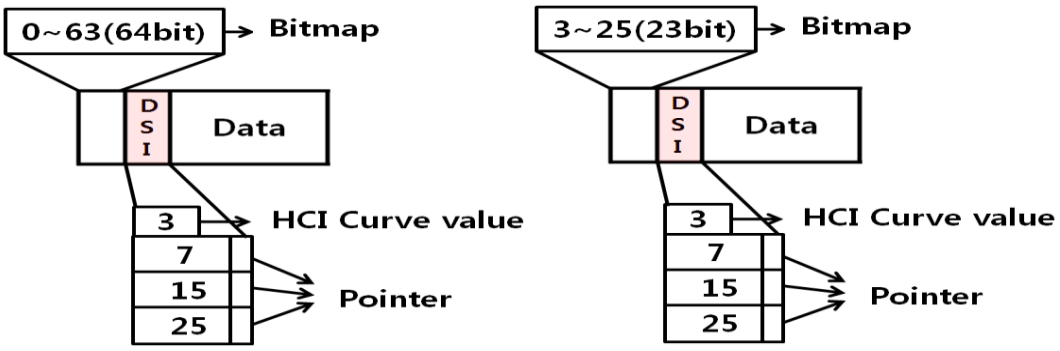


Fig. 3. (a) B·DSI diagram (b) DB·DSI diagram

3. The Proposed Algorithm

In this paper, we propose two algorithms for the optimization of bitmap indexing. The first is the algorithm for the Selective Bitmap-based Spatial Indexing (SBSI) technique. SBSI was developed as a solution to the drawbacks of B·DSI. The second is the algorithm for the Adaptive Bitmap-based Spatial Indexing (ABSI) technique, which facilitates the selective tuning of the bitmap index.

- *IndexB* = BSI index size
- *IndexD* = DSI index size
- DAT = data arrival time
- *k* = number of objects to be searched
- 2^n = order level of DSI index (n = 0, 1,⋯, n)
- *R* = total number of objects that include the last number within the range, in the case of a range query
- *r* = number of objects that exist within the range, in the case of a range query
- *M* = total number of objects that include result values up to the last number within the range, in the case of a *k*NN query

• m = total number of objects that include the last number of the result value object, in the case of a kNN query

3.1 SBSI Scheme

SBSI is a method for improving both the access latency and tuning time. **Fig. 4** shows the structures of the SBSI and B·DSI methods. Firstly, to compare the access latency of SBSI and B·DSI, we will explain the different structures of DSI (**Fig. 4** (a)) and DAT (**Fig. 4** (b)).

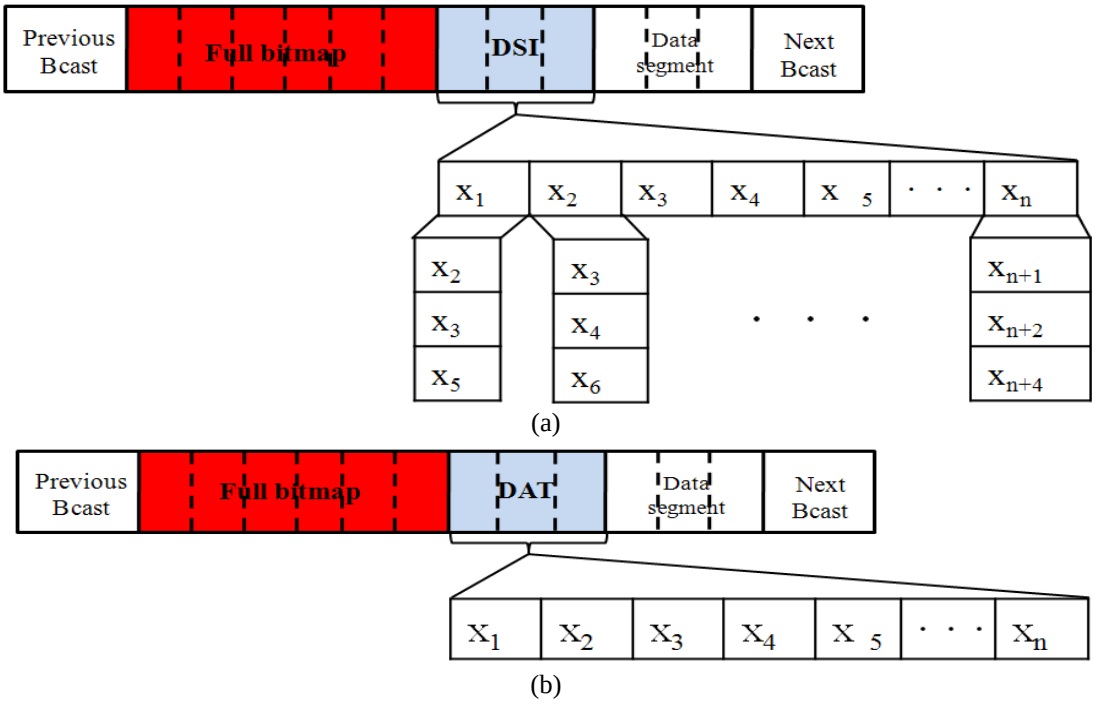


Fig. 4. (a) B·DSI diagram (b) SBSI diagram

In **Fig. 4** (a), we assumed that the indexes of DSI increase by the ratio of 2^2. From the structures of DSI indexes, it is known that they have overlapping values. (e.g., x_1 has x_2, x_3, x_5, and x_2 has x_3, x_4, x_6. That is, x_2 and x_3 are overlapping indexes). In **Fig. 4** (b), DAT has a single value for each data point (the sizes of x_i are all the same), so there are no overlapping indexes. In addition, the access latency can be decreased by decreasing the broadcasting cycle by means of storing only a single value in x_i. To understand the difference in tuning times for SBSI and B·DSI, we compared the structures of the full bitmap, DSI, and DAT. In the case of B·DSI, the full bitmap and DSI is read, whereas SBSI selectively tunes to the full bitmap according to the query request (e.g., step 2 of range query and step 3 of kNN query). DSI tunes to the data through 2^2, whereas DAT selectively tunes to the times of the relevant data by using the result values obtained through the full bitmap to decrease the tuning time. **Fig. 5** shows the SBSI structures in the cases where the data sizes are the same and where they are different. The SBSI method is a technique for finding information on the location of objects and their distances from a query point; such information can be obtained only from the bitmap index. Existing bitmap indexes process queries after reading all the indexes, whereas SBSI defines the scope of the relevant indexes for selective tuning. It is assumed that the size of the data sent

from the server is equal. Therefore, the client can identify the arrival time of objects on the basis of the size and bandwidth of data from a server. Thus, SBSI eventually reduces the tuning time and battery usage of a mobile device.

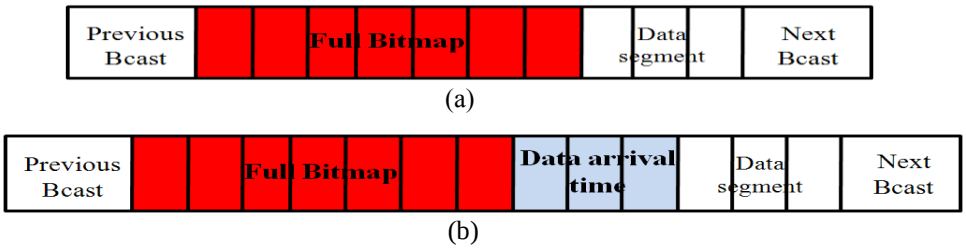


Fig. 5. (a) SBSI diagram for equal data size (b) SBSI diagram for random data size

When the data size varies, information on data arrival time is added. In **Fig. 5** (b), in addition to the bitmap index, the data arrival time element is accessed to obtain the arrival time of the variable. At the same time, we note that all data arrival times can be identical. In this case, by referring to the bitmap index, the client can selectively access the queried object, and it is not always necessary to read the data arrival time. That is, the access latency on the client side can be optimized by selectively accessing both data and data arrival time information.

- Transmission speed (bps) = bandwidth (baud) × data capacity (bits)

The range query processing is as follows.

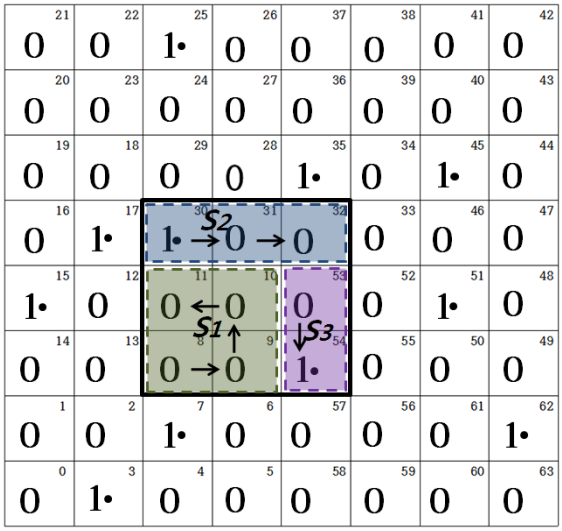


Fig. 6. SBSI range query processing

The algorithm and expression for processing the range query (algorithm 1) is as follows.

- Probe wait $C = \frac{1}{2}(Data+IndexB)$
- Access time $=\frac{R}{2}(Data+IndexB)+C$
- Tuning time $= r(Data+IndexB)+C$

Step 1 is to read the user's query and then to read the bitmap indexes of the candidates (s_i)

located within the range of a given R. As shown in **Fig. 6**, a search is performed only for the s_i that are in the ranges of {8, 11}, {30, 32}, and {53, 54}.

Algorithm 1. Range query processing based on SBSI

Input: query range R;
Output: objects within query range R;
Procedure:
01 : compute the target segments set R, each set s_i
denoted by [s_{in}, s_{out}]; // s_{in}: s_i start point,
s_{out}: s_i end point
02 : begin the initial probe and retrieve bitmap;
target=NULL; result=NULL;
03 : **while** R is not empty **do**
04 : **for** each object s_i covered by R **do**
05 : result=result ∪ {s_i};
06 : **end for**
07 : **end while**
08 : return result;

Step 2 is to compare each s_i with objects that exist within the range (R) and to save the duplicate s_i as result values. After this procedure is repeated, the value of the final result is the output. As shown in **Fig. 6**, in the case of the bitmap index, s_1, s_2, and s_3 read 4, 3, and 2 bits, respectively. This implies that tuning is carried out for a total of 9 bits in order to obtain object information within the requested range. Subsequently, data on objects {30} and {54} that exist within *S* are read before query processing is completed.

The *k*NN query processing is as follows.

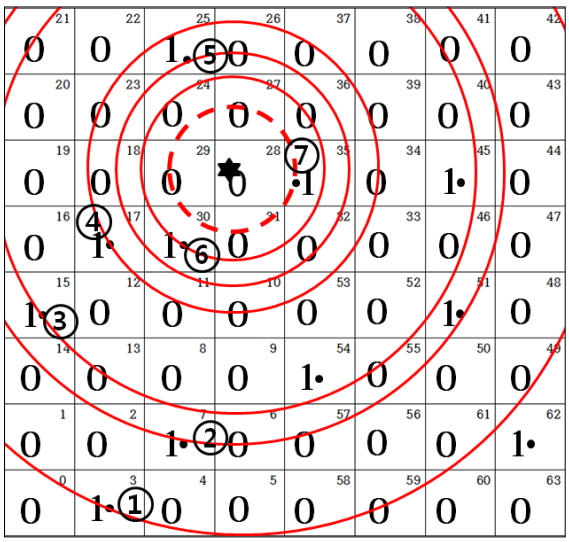


Fig. 7. SBSI *k*NN query processing

The algorithm and expression for processing the *k*NN query (algorithm 2) is as follows.

- Access time = $\frac{M}{2}(Data+IndexB)+C$
- Tuning time = $k(Data+IndexB)+C$

Algorithm 2. *k*NN Query Processing Based on SBSI

Input: a query point, *q*, the number of nearest neighbors, *k*;
Output: *k* nearest neighbors to *q*;
Procedure:
MBC(Minimal Boundary Circle) that contains currently found *k* object;
01 : compute the Hilbert Curve value of *q*;
02 : compute the target segments set *B*, each set b_i
that contains object bit=1;
03 : begin initial probe and retrieve bitmap;
R=∞; Target=NULL; Result=NULL;
04 : **while** k≦*B* **do**
05 : **for** each object b_i covered by *MBC* **do**
06: **if**(*MBC* within b_i) **then**
07 : *KNN'*=*KNN'*∪{b_i};
08 : **else**
09 : continue search;
10 : **end if**
11 : **end for**
12 : *B*=*B*-[b_i];
13 : **end while**
14 : *KNN*=*KNN'*
15 : return *KNN*;

Step 1 is to read the user's query and to find the query point.
Step 2 is to read the bitmap index and to select an object of bit value 1 as the candidate (*B*).
Step 3 is to save the candidate object (O_i) as the result if O_i exists within the *MBC*. The search is not performed if the candidate object exists outside the *MBC*.

After reading up to {3, 7, 15}, a circle with {7} as its center is drawn, where {7} is the second-closest object to the query point. This procedure is repeated until the number of objects within the *MBC* is equal to 2. Then, the result values {30} and {35} are output. Consequently, the bitmap index reads a total of 34 bits, starting from {3} to the last number {36}, which include the result value and are located within the *MBC* range. If there is an object that exists outside the *MBC*, the search can be skipped; this eventually reduces the tuning time for the bitmap index.

3.2 Adaptive Bitmap-based Spatial Indexing (ABSI)

The Adaptive Bitmap-based Spatial Indexing (ABSI) is a method suggested to selectively tune to the full bitmap. To better understand the ABSI, we will explain by taking the example shown in **Fig. 8**. The map in **Fig. 8** is of Greece, and each dot represents the distribution form of its cities and villages. According to this distribution map, server divide the area into units of value δ, and if there are cities or villages in the divided areas, server indicate them using shaded boxes. The adaptive bitmap of **Fig. 8** is 64 bits, and it can be checked if there are objects in the shaded boxes. After the ABSI has been read, the full bitmap can be tuned to selectively. (E.g., in the 64 region of the ABSI of **Fig 8**, the 29 region where the bit value is 0 are skipped (fullbitmap's tune time decrease about 45% by tuning ABSI)).

In **Fig. 8**, objects are placed in a two-dimensional grid using the HC (i.e., an $n \times n$ grid with cells of size δ is formed). Each cell in the grid stores the bit 1 if there is an object in the cell and the bit 0 otherwise. An ABSI that allows fast access to the cells of interest is also created based on the bit information stored in the cells. **Fig. 9** shows the overall grid structure of the ABSI scheme.

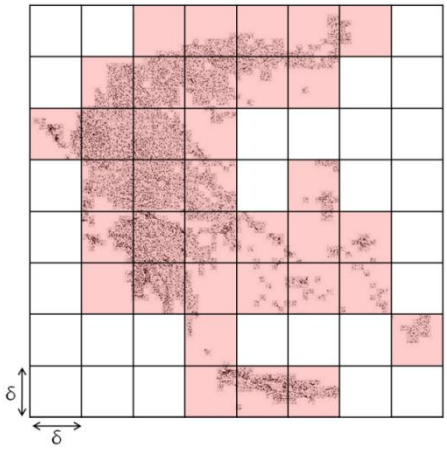


Fig. 8. Example of ABSI diagram

The following packet structure depicts the SBSI scheme employing the adaptive bitmap. This combination of the SBSI scheme and the adaptive bitmap is called the ABSI scheme.

Fig. 9. ABSI diagram

Range query and *k*NN query processing in the ABSI scheme are described below. **Fig. 10** shows range query processing based on the ABSI technique.

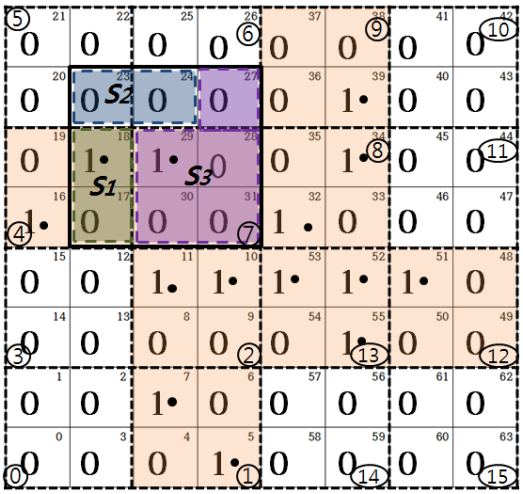


Fig. 10. ABSI range query processing

In the HC map shown in **Fig. 10**, the shaded rectangle (R) indicates the area that the enquirer is trying to locate, and the ranges of the area include (17-18), (23-24), and (27-31). The three dotted-line rectangles (s_i) inside R represent candidates. The algorithms for the range query in **Fig. 10** and the *k*NN query in **Fig. 11** are similar to algorithms 1 and 2, except that the adaptive

bitmap is added.

The steps of range query processing in the ABSI scheme are given below.

Step 1: An ABSI is generated with δ (δ = 2). The generated ABSI is $0110100111001100_{(2)}$. In **Fig. 10**, the Hilbert values of the cells in the ABSI are marked with circled numbers, and those storing the bit 1 are colored in pink.

Step 2: The submitted user query is examined to detect candidates (spatial areas of interest). In **Fig. 10**, the candidates found are {17, 18}, {23, 24}, and {27, 28, 29, 30, 31}.

Step 3: In the full bitmap (the base index data), only the cells that are matched to the 1-bit cells of the ABSI (i.e., ①, ②, ④, ⑦, ⑧, ⑨, ⑫, and ⑬) and to the candidates (i.e., {17, 18}, {23, 24}, and {27, 28, 29, 30, 31}) are accessed. In **Fig. 10**, the matched cells in the full bitmap are {17}, {18}, {28}, {29}, {30}, and {31}. Only these cells are selectively searched.

Step 4: Among the selectively searched cells, i.e., {17}, {18}, {28}, {29}, {30}, and {31}, those that have the bit 1 are output and the query operation is completed. In the example in **Fig. 10**, the output is {18} and {29}.

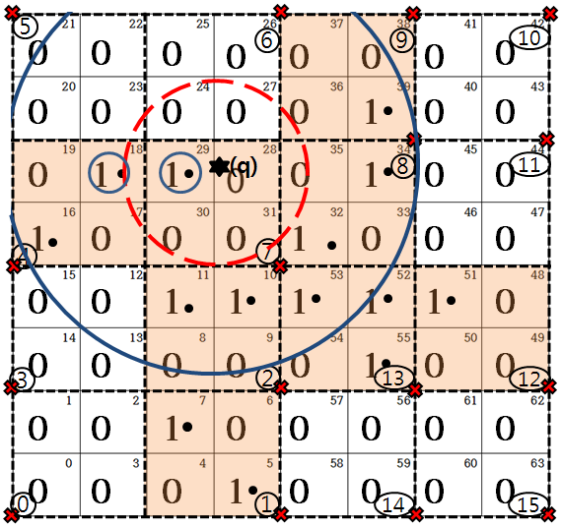


Fig. 11. ABSI 2NN query processing

Fig. 11 shows the *k*NN query processing based on SBSI. The number of objects (*k*) to be searched form the query point (*q*) is assumed to be 2. The circle with the unbroken line indicates 2NN corresponding to the ABSI, while the circle with the dotted line indicates 2NN of the full bitmap. The steps of *k*NN query processing in the ABSI scheme are as follows.

Step 1: The submitted user query is examined to analyze the content and query point. In the example in **Fig. 11**, the Hilbert value of the query point (*q*) is {28}.

Step 2: 2NNs are detected in the ABSI. To find two nearest cells in the ABSI, the distance between the query point and the red "X" mark that represents a point in each adaptive bitmap cell is measured. In **Fig. 11**, the computed 2NNs in the adaptive bitmap are ⑦ and ⑧ Between these two, the one farther from the query point (i.e., ⑧)is selected to form an *MBC*.

Step 3: Among the adaptive bitmap cells, those that are within the range of the blue-colored *MBC* and that store the bit 1 are accessed. In **Fig. 11**, the cells that satisfy these two conditions are ②, ④, ⑦, ⑧, ⑨, and ⑬. In the full bitmap, only the cells that are matched to the adaptive bitmap cells ②, ④, ⑦, ⑧, ⑨, and ⑬ are tuned to find 2NNs.

Step 4: In full bitmap searches, {18} and {29} are discovered as 2NNs. A narrower *MBC* (the red-colored circle in **Fig. 11** is formed with the node farther from the query point, i.e., {18}. The results ({18}, {29}) are returned and the query operation is completed.

4. Experimental Classification Results

The evaluation of the results in this section indicate that the HC order changes with the access latency and tuning time. The simulation model consists of a server, a random number of clients, and a broadcast channel. The size of the range query is defined as 10% of the total map. We performed an experiment with 10,000 iterations in order to carry out an accurate analysis. The HC order, denoted as HC order *N*, varies from HC order 5 (4^5 bits) to HC order 8 (4^8 bits).

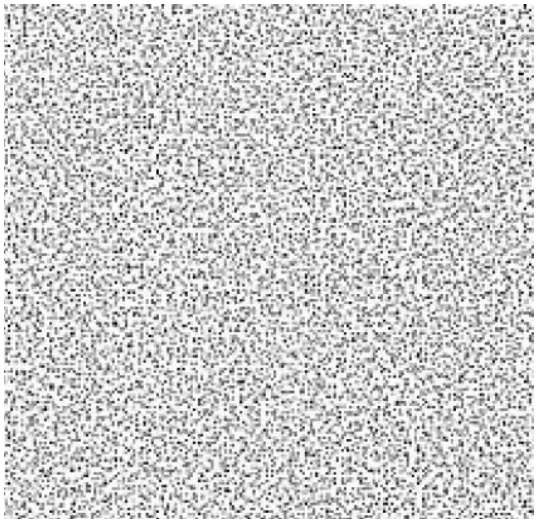
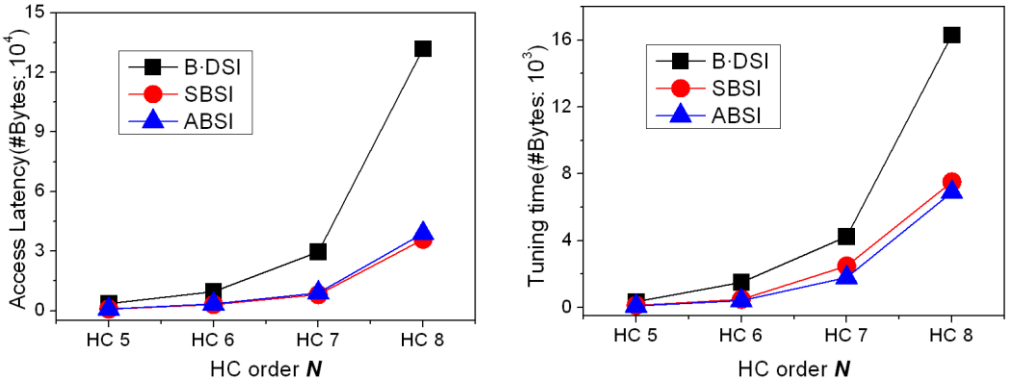


Fig. 12. UNIFORM dataset

This paper shows the experimental results according to the changes in HC order, to verify the efficiency of the proposed bitmap structure. The selective tuning of the bitmap index helped to check whether a given object exists and to measure distances. Thus, the tuning time and access latency was reduced. In the UNIFORM dataset of **Fig. 12**, HC order *N* (4^n/10) points are uniformly generated in a square Euclidean space. In the experiment, the x-axis shows variable and the y-axis, time. The reason why the y-axis is labeled with bytes is that the transmission time varies depending on bandwidth so it is general to put bytes. The size of a data object is set to 10 bytes. For each DSI index of 2 bytes, the BSI index is 4^n bits. In addition, the reason that we divided NN and *k*NN to conduct the experiment was to measure the changes in the bitmap by using the HC order as a variable in the case of NN. On the other hand, in the case of *k*NN, we did not use the HC order as a variable, but fixed it as HC 7 and then used the variables of *k*, *m*, and data ratio to measure the experimental results.

4.1 Range Queries

We compared the performance of the techniques in the range query operation on the basis of the evaluated indexes. First, we fixed the size of the range query in order to evaluate the performance in range query processing.



(a) Access Latency (Range) (b) Tuning time (Range)

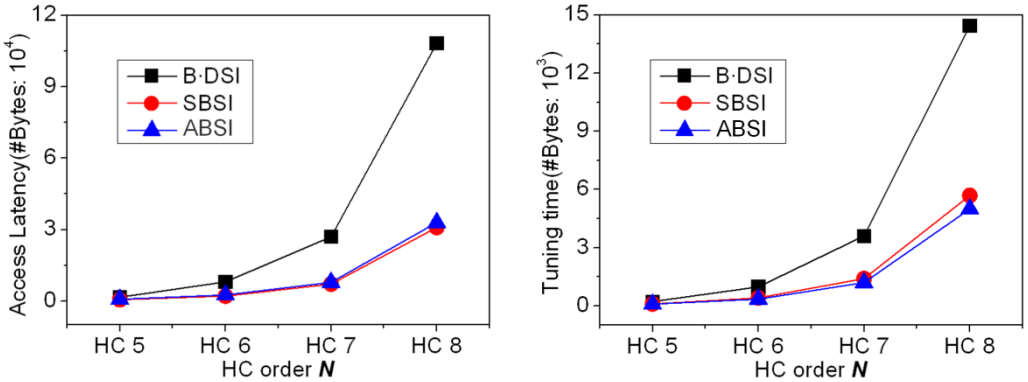
Fig. 13. Range query

Fig. 13 (a) shows that SBSI is superior to B·DSI and ABSI. As the HC order size increases, in the case of access latency, the inclinations of SBSI and ABSI increase more gradually than that of B·DSI. In the range query, SBSI generally outperforms B·DSI in terms of access latency. The range query processing is as follows.

In the case of B·DSI, the access latency is greater because of the increased slope; this can be attributed to the increase in the HC order. Because of the HC order, the total size is the same as the bitmap index size. In addition, **Fig. 13** (a) shows that the ABSI includes an adaptive bitmap index, and hence, the index size is large. However, the access latency cannot be improved by data tuning. **Fig. 13** (b) shows that the ABSI scheme that allows selective access using the adaptive bitmap yields the best (lowest) tuning time.

4.2 Nearest Neighbor Queries

We compared the performance of the techniques in the NN query on the basis of the evaluated indexes. First, we fixed the NN queries, and then evaluated the performance in processing NN queries. **Fig. 14** (a) shows that SBSI is superior to B·DSI and ABSI in terms of access latency. As the HC order size increases, the inclinations of SBSI and ABSI increase more gradually whereas the inclination of B·DSI increases significantly.



(a) Access Latency (NN) (b) Tuning time (NN)

Fig. 14. NN query

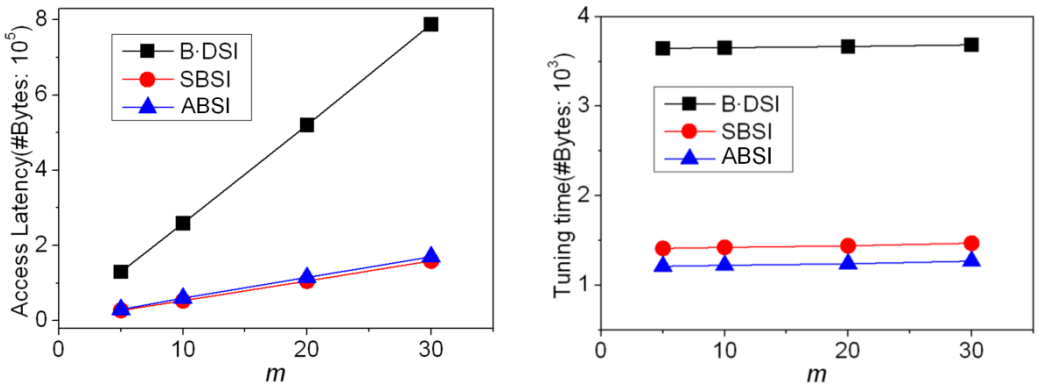
Fig. 14 (b) shows that ABSI outperforms the other techniques in terms of tuning time. The results for the tuning time in the case of ABSI and SBSI are similar. Because of the reading of

the adaptive bitmap, the optimal tuning time can be obtained. However, the tuning time of B·DSI becomes larger and the slope is visibly increased.

4.3 *k*-Nearest Neighbor Queries

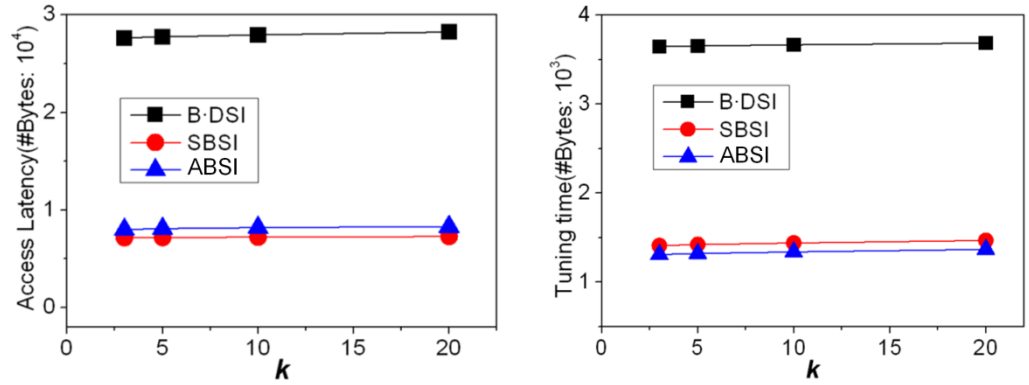
In order to evaluate the performance for *k*NN queries, 10NN queries were examined while varying the number of index *m*. **Fig. 15** shows the simulation results of the UNIFORM dataset. As shown in **Fig. 15** (a), ABSI has a slightly higher access latency than SBSI. This is because the adaptive bitmap is added, so that ABSI can selectively tune to the entire bitmap. However, since the adaptive bitmap has additional but smaller-sized information, the access latency do not differ significantly. On the other hand, the tuning time is improved more in ABSI than in SBSI.

Fig. 15 (b) shows the server's tuning time. The ABSI scheme that utilizes the index information in the adaptive bitmap is better than SBSI and B·DSI. The ABSI allows unnecessary index entries to be skipped during a search. This reduces the overheads in spatial query processing, which increase significantly as the index size increases.



(a) Access Latency (10NN) (b) Tuning time (10NN)

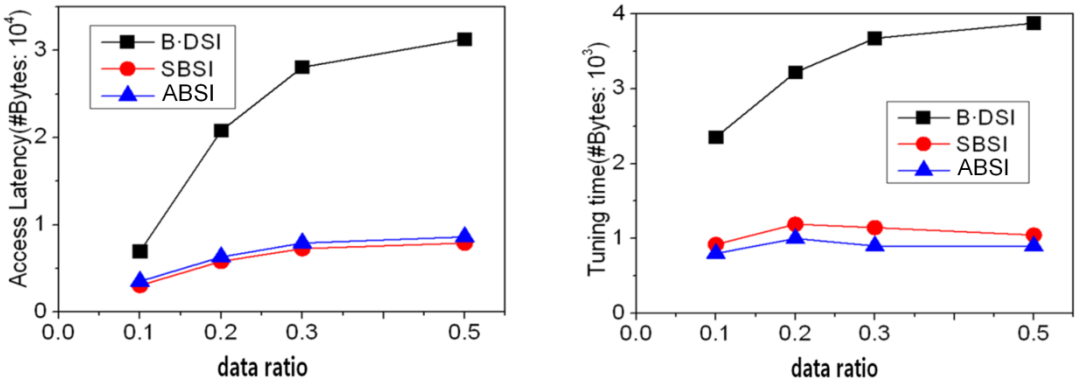
Fig. 15. Performance of *m* number



(a) Access Latency (*k*NN) (b) Tuning time (*k*NN)

Fig. 16. *k*NN queries

Fig. 16 shows the performance (access latency and tuning time) of *k*NN queries for varying *k*. The SBSI scheme shows the smallest access latency, as shown in **Fig. 16** (a). In terms of the tuning time, the **ABSI** scheme shows the best performance (see **Fig. 16** (b)).



(a) Access Latency (10NN) (b) Tuning time (10NN)

Fig. 17. Performance of data ratio

Fig. 17 shows the performance for 10NN queries for the case in which the data ratio was varied (data ratio indicates the data distribution ratio according to which the data frequency existing in the map changes. According to these data ratios, the numbers of data inserted correspond to 10%, 20%, 30%, and 50% of the numbers of the HC order). The HC order was 7. Naturally, the amount of data to be read increases as the data ratio increases. As shown in **Fig. 17** (a), the access latency in the B·DSI scheme grows significantly as the data ratio increases. On the other hand, the SBSI and ABSI schemes show relatively smooth curves that increase slowly as the data ratio increases. The performance gap between SBSI (or ABSI) and B·DSI will grow further as the number of objects increases. This result indicates that the proposed schemes are effective methods for supporting LBSs with a large amount of spatial data. In **Fig. 17** (b), the tuning time in the B·DSI scheme increases significantly as the data ratio increases. In the SBSI and ABSI schemes, changes in the tuning time with increasing data ratio are small. Note that in the SBSI and , ABSI schemes, the tuning time is slightly shorter when the data ratio is 0.3 or 0.5 than when it is 0.2. This is because the search range of the 10NN query decreases as the grid density (the number of cells along the axes of the grid) increases, which, in turn, decreases the Hilbert values used to answer the submitted query. The experimental results showed that the proposed schemes have a performance advantage over the conventional schemes in both range queries and *k*NN queries.

5. Conclusion

In this paper, we have suggested solutions to problems encountered in indexing schemes. Existing indexing schemes involve reading data that are not essential, or obtaining a greater number of indexes than necessary, which increases the access latency. The SBSI scheme and the ABSI have been proposed and explained as solutions that can be adopted in wireless environments. In the case of SBSI, only the bitmap was used in order to reduce the index size. While the whole bitmap index was inserted, the tuning time was reduced because of selective tuning achieved by using the bitmap index. The selective reading of the bitmap index helps to check whether a given object exists and to measure distances. That is, this scheme enables the client to access bitmap index information selectively, thereby reducing access latency on the client side. The proposed ABSI can improve the query processing time in the SBSI scheme. By referring to the information in the ABSI, the proposed scheme can perform query operations without accessing the 0-bit entries in the bitmap index that have no object. This increases the efficiency in spatial query processing. If the client's battery energy is big enough

not to be affected by energy consumption, SBSI is suitable when fast query processing is required. Meanwhile, in case the client's battery energy is so small that energy consumption should be minimized, ABSI shall be appropriate. The proposed method deals with the problems of wireless transmission errors. Even though any error occurs during wireless transmission, as long as bitmap and data arrival time (DAT) are recognized, the order and location of all the objects shall be checked and the data arrival time shall be identified so that it can be a solution for wireless transmission error. That is why DSI was modelled in communication environment. In this study, we considered cases in which the object was fixed. The future direction of study will be the development of an optimal indexing scheme that can be applied in the case of moving objects.

References

[1] B. Rao, L. Minakakis, "Evoluation of Mobile Location-based Services," *Communications of the ACM*, vol. 46, no. 12, pp. 61-65, Dec. 2003. Article (CrossRef Link)

[2] P. Bellavista, A. Kupper, S. Helal, "Location-based Services: Back to the Future," *IEEE Pervasive Computing*, vol. 7, no. 2, pp. 85-89, Apr.-June, 2008. Article (CrossRef Link)

[3] J. Xu, X. Tang, W.-C. Lee, "Time-critical On-Demand Data Broadcast : Algorithms, Analysis, and Performance Evaluation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 1, pp. 3-14, Jan. 2006.

[4] D. Aksoy, M. Franklin, "RxW: A Scheduling Approach for Large-Scale On-Demand Data Broadcast," *IEEE Transactions on Networking*, vol. 7, no. 6, pp. 846-860, December, 1999. Article (CrossRef Link)

[5] K. Park, H. Choo, V. Patrick, "A Scalable Energy-Efficient Continuous Nearest Neighbor Search in Wireless Broadcast Systems," *Wireless Networks*, vol. 16, no. 4, pp. 1011-1031, 2010. Article (CrossRef Link)

[6] S. Acharya, R. Alonso, M.J. Franklin, S.B. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communications Environments," in *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, vol. 24, no. 2, pp. 199-210, May 1995. Article (CrossRef Link)

[7] B. Zheng, D.L. Lee, "Information Dissemination via Wireless Broadcast," *Communications of the ACM*, vol. 48, no. 5, pp. 105-110, May 2005. Article (CrossRef Link)

[8] K. Michael, G. Damianos, M. Aristides, "A Mobile Tourism Recommender System," *IEEE Computers and Communications*, pp. 840-845, June 2010. Article (CrossRef Link)

[9] Y.-H Chang, "A Graphical Query Language for Mobile Information Systems," *ACM SIGMOD Record*, vol. 32, no. 1, pp. 20-25, March, 2003. Article (CrossRef Link)

[10] T.H.N. Vu, K.H. Ryu, N. Park. "A Method for Predicting Future Location of Mobile User for Location-Based Services System," *Computers & Industrial Engineering*, vol. 57, no. 1, pp. 91-105, July 2009. Article (CrossRef Link)

[11] B. Zheng, W.-C. Lee, K.C.K. Lee, D.L. Lee, M. Shao, "A Distributed Spatial Index for Error-Prone Wireless Data Broadcast," *Very Large Data Bases Journal*, vol. 18, no. 4, pp. 959-986, Aug. 2009. Article (CrossRef Link)

[12] S. Qiao, C. Tang, J. Peng, H. Li, S. Ni, "Efficient k-Closest-Pair Range-Queries in Spatial Database," *Web-Age Information Management*, pp. 94-104, July 2008.

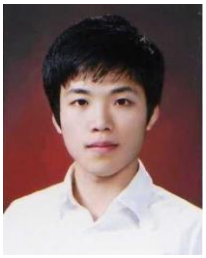
[13] W.-S. Ku, R. Zimmermann, H. Wang, "Location-Based Spatial Query Processing in Wireless Broadcast Environments," *IEEE Transactions on mobile computing*, vol. 7, no. 6, pp. 778-791, June 2008. Article (CrossRef Link)

[14] H. Sin, M. Lee, J. Choi, S. Lee, "Bitmap-based Spatial Index on Wireless Broadcast," *SIGDB-KISS*, vol. 23, no. 1, pp. 23-36, Apri. 2007.

[15] B. Zheng, W.C. Lee, D.L. Lee, "Spatial Queries in Wireless Broadcast Systems," *Wireless Networks,* vol. 10, no. 6, pp. 723-736, Nov. 2004. Article (CrossRef Link)

[16] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," in *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, pp. 47-57, June 1984. Article (CrossRef Link)

[17] S. Berchtold, D. A Keim, H.-P Kriegel, "The X-tree: An Index Structure for High-Dimensional Data," in *Proc. of Intl. Conf. on Very Large Data Bases*, pp. 28-39, 1996.
[18] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," in *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, vol. 19, no. 2, pp. 322-331, 1990. Article (CrossRef Link)
[19] T. Sellis, N. Roussopoulos, C. Raloutsos, "The R+-Tree: A Dynamic Index for Multi- Dimensional Object," in *Proc. of Very Large Data Bases*, pp. 507-518, 1987.
[20] J. Zheng, M. Zhu, D. Papadias, "Location-Based Spatial Queries," in *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, pp. 443-454, June 2003. Article (CrossRef Link)
[21] Ken C. K. Lee, W.-C. Lee, H.V. Leong, B. Unger, B. Zheng, "Efficient Valid Scope for Location-Dependent Spatial Queries in Mobile Environment," *Journal of Software,* vol. 5, no. 2, pp. 133-145, Feb. 2010. Article (CrossRef Link)
[22] T. Imielinski, S. Viswanathan, B.R. Badrinath, "Energy Efficient Indexing on Air," in *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, vol. 23, no. 2, pp. 25-36, June 1994. Article (CrossRef Link)
[23] K. Park, H. Choo, "Energy-Efficient Data Dissemination Schemes for Nearest Neighbor Query Processing," *IEEE Transactions on Computers*, vol. 56, no. 6, pp. 754-768, June 2007. Article (CrossRef Link)
[24] T. Imielinski, S. Viswanathan, B. R. Badrinath, "Data on Air: Organization and Access," *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 3, pp. 353-372, May-June 1997. Article (CrossRef Link)
[25] C. Gotsman, M. Lindenbaum. "On the Metric Properties of Discrete Space-Filling Curves," *IEEE Transactions on Image Processing*, vol. 5, no. 5, pp. 794–797, May 1996. Article (CrossRef Link)
[26] P. O'Neil, G. Grafe. "Multi-Table Joins Through Bitmapped Join Indices," *ACM SIGMOD Record*, vol. 24, no. 3, Sep. pp. 8-11, 1995. Article (CrossRef Link)
[27] C.-Y. Chan, Y.E. Ioannidis, "Bitmap Index Design and Evaluation." in *Proc. of Special Interest Group on Management Of Data*, vol. 27, no. 2, June 1998. Article (CrossRef Link)
[28] D. Song, K. Park, "An Efficient Bitmap-Based Selective Tuning Scheme for Spatial Queries in Location-Based Service," in *Proc. of Intl. Conf. on Internet*, pp. 503-509, Dec. 2010.



Doohee Song received a B.S. degree from Wonkwang University of the Department of Electrical Electronics and Information Engineering, Korea, in 2009. He is currently a M.S. student in the Department of Electrical Electronics and Information Engineering, Wonkwang University. His research interests include spatial indexing, data broadcasting, mobile computing.

Kwangjin Park received the PhD degree in computer science from Korea University in 2006. He was a postdoctoral fellow in the Atlantic Data Systems (Atlas) Research Group at the Institut National de Recherche en Informatique et en Automatique (INRIA)-Rennes and located at the Laboratoire dinformatique de Nantes-Atlantique (LINA), Nantes. In 2008, he joined the Schools of Electrical Electronics and Information Engineering, Wonkwang University, where he is an assistant professor. His research interests include spatiotemporal databases, mobile databases, and data dissemination.