Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 04, April-2020

An Efficient Network Coding-based Air Index for Queries in Road Networks

M. Veeresha[1], M. Sugumaran[2], D. Sandeep[3]
Associate Professor[1], Professor[2], Assistant Professor[3]
Department of Computer Science and Engineering[1,2,3],
Santhiram Engineering College[1,3], Nandyal, India.
Pondicherry Engineering College[2], Pondicherry,India

***Abstract*—Wireless broadcast is an efficient strategy for disseminating data to the clients in road networks. *Network Partition Index* (NPI) doesn't address how to handle various data objects are increased to a large number in broadcast system and link errors. To address these problems, an NPI based air index, called Network Coding-based *Spatial Air Index* (NCSAI) has been proposed by using *Hybrid Broadcast* (HB) scheduling and *adaptive XOR-based network coding* for spatial queries in road networks. We conducted various experiments for evaluating query performance. The experimental results show that NCSAI has better query performance compared with state-of-the-art NPI.**

Keywords—Wireless broadcast, network coding-based spatial air index, hybrid broadcast, adaptive XOR-based network coding, spatial queries, road networks.

1. INTRODUCTION

Due to rapid advances in both wireless communication and mobile computing, location-based services are more useful to the clients in real-time and day-to-day applications. Recently, most of the GPS (i.e., Geographical Positing System) enabled devices such as smart phones, laptops and PDAs request nearest information via spatial queries in road networks. The road network has been modeled into time-independent and time-dependent and computed travelling time based on the distance and traffic in a network respectively. However, these models have been suffered from scalability and security [1], [2], [3], [4]. To address these issues, wireless broadcast strategy has been adopted [5]. Wireless broadcast strategies are classified into push-based, pull-based and hybrid scheduling. However, these strategies have been suffered from sequential data access [6], [7]. More research has been done on query processing in Euclidian space rather than road networks [8], [9]. However, in most of the real-time applications query processing in road networks is essential. Air index adopted for shortest path queries in a road network and doesn't address range queries and *k*NN queries [10]. Recently, NPI based air index adopted for various spatial queries in road networks. However, NPI doesn't addressed well scheduling strategy when the data objects are increased to a large number in broadcast system and link errors [11]. To address these problems, we propose NCSAI using HB scheduling and adaptive XOR-based network coding for spatial queries in road networks and improve query performance. The overall idea of this work is explained as follows. The original road networks may contain various data objects such as restaurants, shopping-malls, hospitals, gas-stations and schools etc., and these are classified into two data sets namely general and specific based on client's requirement, and computed cut-off point [12]. Using grid partition strategy, road network is partitioned into small grid cells, and pre-computed some general information such as diameter of each cell and minimum/maximum network distance between every pair of cells that will be carried by NCSAI. At server site, we incorporate adaptive *XOR-based network coding* into broadcast scheduling program, and broadcast general and specific data objects of NCSAI with network connectivity information of each cell using HB scheduling. On the client site, once a client receives a query request from a mobile user then it searches query result in its cache, if found then it return the result. If not found, then the client tunes into channel, retrieves required information and decode the data. Finally, process the query using Dijkstra's shortest path algorithm. If required data doesn't broadcast while tuning then the client send query request to server via pull-based scheduling. While broadcasting, if any free time-slots exist then server broadcast encoded form of client's requested data objects (i.e., specific data objects) along with general data objects based on longest waiting time in the queue on priority order, and updates optimal cut-off point.
The contribution of this paper can be summarized as follows:

- Propose NCSAI using HB scheduling and adaptive XOR-based network coding for spatial queries in a road networks.
- Propose search algorithms at client-site, which includes range query, *k*-nearest neighbor query and continuous *k*-nearest neighbor query.
- Experiments are conducted using real road networks data and compared the performance of NCSAI with state-of-the-art NPI.

The rest of this paper is organized as follows. Section 2 describes brief overview and previous works of both wireless broadcast and network coding, Section 3 presents NCSAI broadcast strategy for spatial queries in road networks, Section 4 discuss various searching algorithms, Section 5 reports performance evolution, and Section 6 concludes the research paper.

2 RELATED WORKS

2.1 Wireless Data Broadcast

Wireless broadcast is an efficient strategy for disseminating various data objects to clients via wireless channel. Various interleaving strategies has been adopted for disseminating air index data to clients, among these (1, *m*) data interleaving strategy widely used in wireless environments is shown in Fig.1.In this strategy, data is divides into *m* equal data segments and broadcast each data segment preceded by air index via wireless channel [5]. For evaluating performance, (1, *m*) data interleaving strategy has been used two performance parameters and these are tuning time and access latency. Tuning time defined as amount of time spends by a client tune into channel and retrieves required data. Access latency defined as amount of time spends by a client to submission of a query and gets query result.

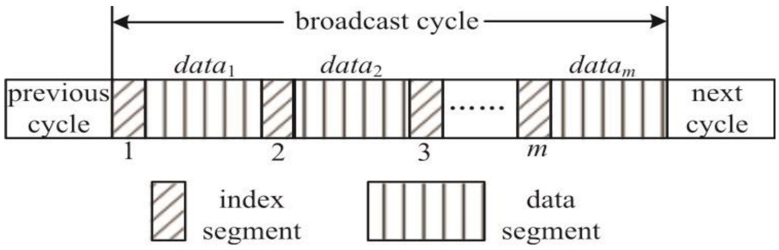


Fig.1. (1, *m*) interleaving strategy

2.2 Query Processing in Wireless Broadcast Environment

In wireless broadcast environment, various indexing structures such as R*-tree, D-tree, Grid-index, Hilbert Curve Index (HCI) and Broadcast Grid Index (BGI) are adopted for processing spatial queries in Euclidian space [13], [14], [15], [16], [17]. However, these indexing structures are not suitable for spatial queries in road networks. In road networks, air indexing structures has been adopted for spatial queries and supports only shortest path queries [18], [19].

2.3 Network Coding

In wireless environment, while disseminating data to clients, some of the data is lost due to unreliable nature of wireless links. In order to address this problem, *Automatic Repeat Request* (ARQ) has been adopted. In ARQ, after each transmission, clients send feedback information to the server, be it a positive or negative acknowledgement. However, ARQ is not efficient if there is an increase in the size of the network. The alternative strategy to address this problem is referred as *forward error control with channel coding*, in which the server broadcasts coded packets (i.e., data packets are coded using network coding schemes) to clients, and then clients perform decoding on coded packets and conclusively get the original data packets. The noteworthy advantages of this strategy are scalability and throughput improvements and robustness of the networks. Network coding has been widely used in wireless multicast environment for improving query performance. Network coding has been primarily introduced for improving performance in multicast routing [20]. Network coding is regarded a general strategy in packet routing that allows an intermediate router to encode an outgoing packet by mixing multiple incoming packets appropriately. In a multi-hop wireless network, to meet unicast demands, game theoretic framework is introduced using network coding [21]. A cross-layer optimization framework is formulated to derive the maximum throughput for multicast traffic. In a random network, the optimization strategy considers network coding in network layer and conflict-free transmission schedules in MAC layer [22]. In a wireless network, one-hop opportunistic network coding is adopted and argued for making rate control and scheduling network-coding aware [23]. The server disseminates a set of data blocks over a broadcast channel to a set of caching clients called index coding problem [24]. The index coding problem is motivated by various applications in wireless networking and distributed computing [25]. For instance, one can find such applications in satellite communication networks where the clients have limited storage and limited maintenance of the received information. Efficient index codes are useful for the wireless network architectures that utilize the network coding and opportunistic listening techniques [26]. Feedback-based adaptive network coding schemes are designed for minimizing decoding delay in each transmission of the packet-based erasure networks [27]. All traditional scheduling algorithms in data broadcast are based on the assumption that only clients requesting the same data item can be satisfied in one broadcast tick. Coding problem is adopted in on-demand broadcast for reducing accessing time. In this approach, each broadcast data item is encoded from utmost two data items and doesn't fully utilize the advantages of data broadcast and coding [28]. However, it is found in this approach that no restriction is considered on the number of encoded data items [29][33][34]. Approaching differently in a proposed frame work where in a grid cell, availability of data objects in each adjacency cell can be encoded and as such it is a significant factor by which different performance objectives can be achieved.

3. NETWORK CODING-BASED SPATIAL AIR INDEXE FOR QUERIES IN ROAD NETWORKS

NCSAI is very useful in wireless broadcast environment for processing spatial queries in road networks as it derives the advantages of network coding. Based on the study vide [11], the partition of the road networks into small grid cells, pre-

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 04, April-2020

computing of the diameter of cell and border point, and formation of NCSAI is shown below in Fig.2. Furthermore, this section describes 1) data segment; 2) adaptive XOR-based network coding and 3) NCSAI broadcasting method on wireless channel.

3.1 Data Segment

Broadcast program contains two major components. They are air index and data. An air index represents NCSAI and data represents data objects respectively. In this work, let us assume that structure of road network is captured by adjacency lists, and each client has unique ID (i.e. identification). From the Fig.2, the grid cell C_2 contains three clients i.e., v_1, v_3 and v_4, and two data objects O_3 and O_4 respectively. In this cell, the adjacency lists of all the vertices are broadcast based on the order of the clients IDs. For example, consider the structure of v_1 adjacency list, it contains two neighbors v_3 and v_4, and each neighbor is represented by four-tuples i.e., (v_3, 1, 1, O_4), where the first field ”v_3” indicates adjacent node i.e.,v_3 connected to v_1, the second field ”1” indicates a weight of an edge (v_1, v_3), the third field ”1” indicates the number of data objects lying on this edge, and the fourth field is a pointer which points to the first data object on this edge. If there is no object on this edge, the pointer is set to *null*. The overall structure of grid cell C_2 is shown in Fig.3.

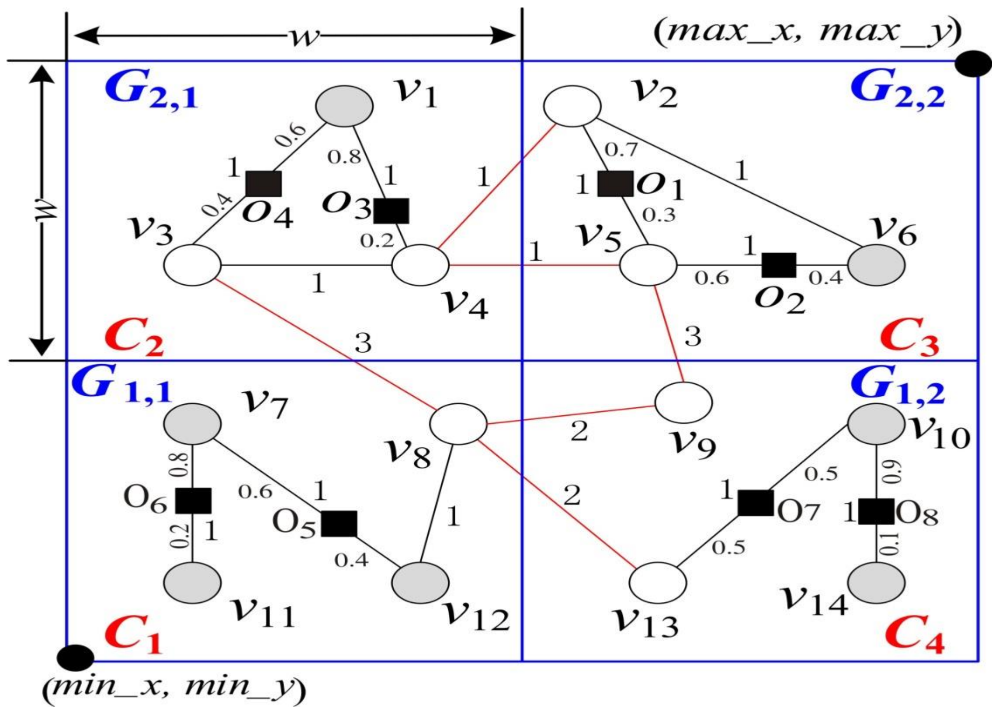
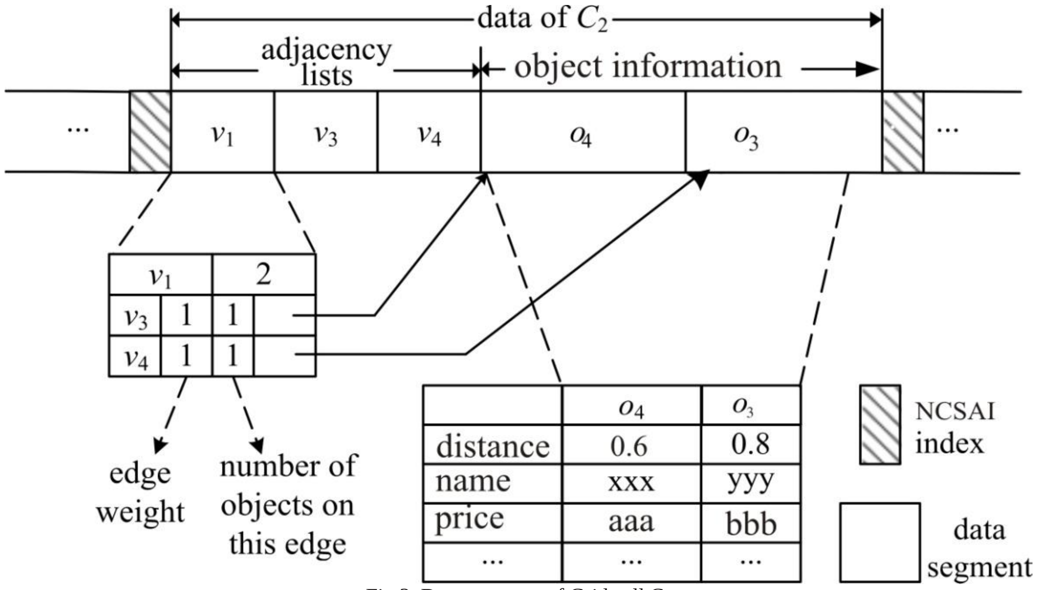


Fig.2. Road Network Partition into 2 × 2 Grid cells



	o_4	o_3
distance	0.6	0.8
name	xxx	yyy
price	aaa	bbb
...	...	...

Fig.3. Data structure of Grid cell C_2

3.2 Adaptive XOR-based Network Coding

In grid cell C_2, we should consider the structure of v_1 adjacency list. It contains two neighbors' v_3 and v_4. v_1 to v_3 contains data object O_4 and v_1 to v_4 contains data object O_3. For example, v_1 to v_3 contains one or more than one data objects; then these data objects are encoded using adaptive XOR-based network coding and broadcast. At client site, decoding takes place. NPI doesn't satisfy multiple client requests in a single tick as network coding concept has not been used. NCSAI, being a more flexible concept satisfies multiple client requests in a single tick using adaptive XOR-based network coding. The salient futures of the concept are stated here. 1) Adaptive XOR-based network coding is a simple one with fewer overheads; 2) Server should know what type of data objects are being requested by clients as well as the clients' cached data;
3) In a grid cell, based on the availability of data objects in a adjacency cell, data objects are encoded using adaptive XOR-based network coding and broadcast by the server and 4) Server and client's site use simple *XOR* operation for encoding and decoding the data.

3.3 How to broadcast NCSAI with data segment on Wireless channel

The server broadcasts NCSAI with data segment via wireless channel using (1, *m*) data interleaving strategy. Then the interested client tunes into channel, retrieve the data and decode it.

4. PROCESSING OF SPATIAL QUERIES AT CLIENT SITE USING NCSAI

This section describes various searching algorithms at client site which include range query, *k*-nearest neighbor query, and continuous *k*-nearest neighbor query. The objective of these algorithms is to process the query efficiently and return result to the client with the help of NCSAI. For processing queries in road networks, Dijkstra's shortest path algorithm has been followed because it is simple and efficient for small sub-graph retrieved and decoded by clients. In this task, it is assumed that the clients' locations are located at network nodes and easily extended to support the case where the clients' locations are put up along the network edges [4], [18], [19].

4.1 Range Query

In road networks, range query retrieves all data objects within a network distance d from a query point q, and it is denoted as $(q, d, S) = \{o \mid o \in S \wedge \|o, q\| \leq d\}$, where q is a query point, o is a data object, d is a network distance and S is a dataset in road networks.

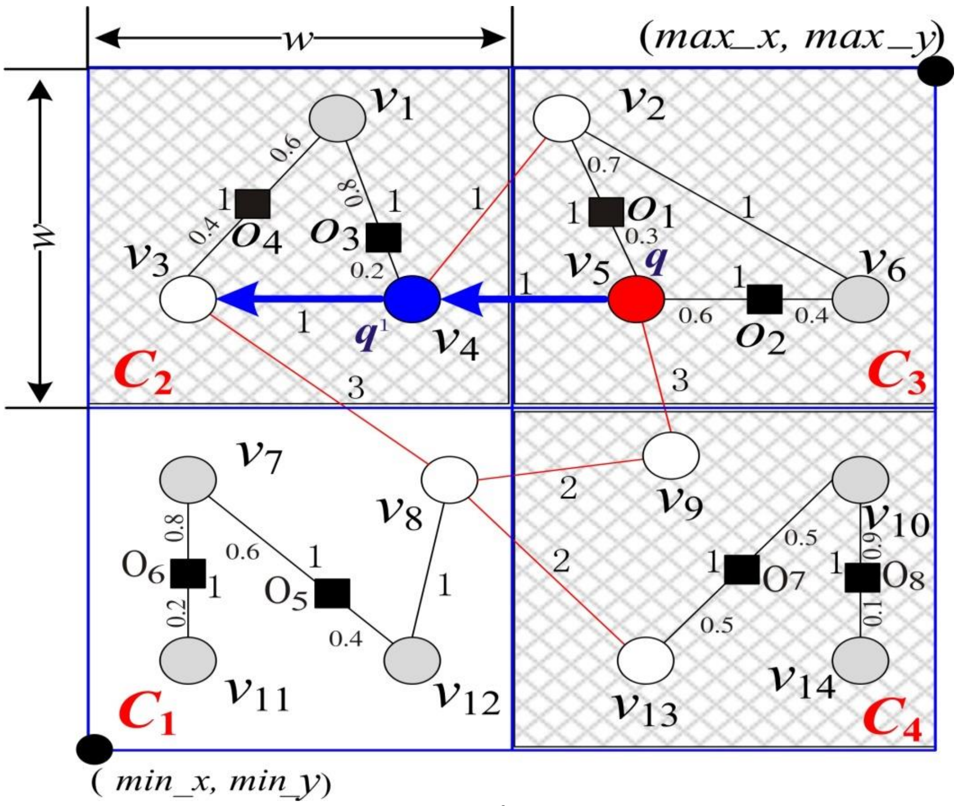


Fig.4. Processing of Range Query

Algorithm 1: Range Query

Procedure Range_query(q, d, S);
Input: q – query point, d – distance, S – data set
Output: valid data objects within the range d
▷% receive_query( ) – a client waits for receiving a query from the mobile user, perform encoding and decoding at server and client sites respectively using *XOR*-based network coding, t – time, C_q– valid grid cells from a query point q, R_q– valid range query result from a query point q, $\alpha_{q,i}$ – minimum or maximum network distance from a query point q, retriveIndexHeader( ) – the client retrieves index header, SleepUntilCellBroadcast( ) – client sleeps until required data broadcast, retriveCell – client retrieves the cells containing query result, adjacencyLists – adjacency lists are forms as sub graph, Dijkstra(subGraph, q, d) – process the query using Dijkstra's shortest path algorithm and return query result; compute a cut-off point *cp* using [12] %
▷% perform encoding and decoding at server and client sites respectively using adaptive XOR-network coding %

```
begin
1:  divide the data set $S$ into general and specific;
2: compute a cut-off point cp;
3: query = receive_query( );
4: if query results found at client cache then
5:    return result;
6: else
7:      Listen_channel( );
8:       return result;
9: end if
10: end

1: Procedure Listen_channel( );
2: begin
3: wait $t$ seconds for required data;
4:  if a client's  required data doesn't broadcast then
5:       client sends query to server;
6:       for each time slot do
7:          if empty slot exist then
8:             select specific data objects based on waiting time and priority;
9:          end if
10:       end for
11:      update cut_off point cp;
12: else if required data is received from the server then
13:      NCSAIHeader  =  retriveIndexHeader( );
14:      find out the grid cells $C_q$ containing query $q$;
15:      read the $q^{th}$ row $R_q$ corresponding to $C_q$ in the matrix;
16:       for each cell $C_i$ do
17:          if $\alpha_{q,i} <= d$ then add $C_i$ to the candidate cell
18:             sort the candidate cells by their arrival times;
19:              for each candidate cell do
20:                 SleepUntilCellBroadcast( );
21:                 Listen_channel( );
22:                 adjacencyLists = retriveCell( );
23:                 subGraph.add(adjacencyLists);
24:                 return Dijkstra(subGraph, $q$, $d$);
25:             end for
26:          end if
27:      end for
28:  end if
29: end
```

In the light of the above mentioned algorithm, it may be assumed that the client being located at node v_5 requests range query to find "four restaurants from query point q with network distance d = 3". Then the range query result obtained is O_1, O_2, O_3 and O_4.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 04, April-2020

4.2 k-Nearest Neighbor Query

K-Nearest Neighbor Query defined as a mobile client retrieves *k*-nearest data objects from a query point *q*. Like a range query, the range is not fixed in a *k*-nearest neighbor query and it will be depend on the value of *k* and location of query point *q*. In this work, we estimate network distance d_{max} based on NCSAI in order to find candidate cells. Once d_{max} is estimated then *k*-nearest neighbor query is converted into range query and retrieve candidate data objects within a network distance d_{max} from a query point *q*. The $d_{max}$is estimated based on the upper bounds of network distance between any object in C_u and any objects in C_v and denoted as $UB\ (C_u, C_v)$.

***Lemma*:** Given two cells C_u and C_v, the network distance between any object *u* in C_u and any object *v* in C_v is bounded by $(diameter(C_u)+diameter(C_v)+\beta_{u,v})$ and it is denoted as $UB(C_u,C_v) \leq\ diameter(C_u)+diameter(C_v)+\ \beta_{u,v}$.

Based on above lemma, d_{max} is estimated as follows. For example a query point *q* is located in cell C_q, we access C_i based on the non-descending order of $\alpha_{q,i}$ that is C_i is visited earlier than C_j if $\alpha_{q,i} < \alpha_{q,j}$ there is tie, $UB(C_q,C_i)$ and $UB(C_q,C_j)$ are used as tie-breaker and one with smaller *UB* value will be visited first. In this process two parameters are used, one is *count* i.e., the total number of data objects in all the cells visited so far, and the other is $UB(d_{max})$ for each visited cell C_i. We calculate $UB(C_q,C_i)$ and set $UB(d_{max})$ to the maximum $UB(C_q,C_i)$ found so far and mean while we check $|C_i|$, and update count. This process is continuing until *count* reaches to value *k.*

Algorithm 2: *k*NN Query

```
Procedure kNN _query(q, k, S);
Input: q - query point, k – an integer, S - data set
Output: valid k nearest data objects from query point q
▷% receive_query( ) – a client waits for receiving a query from the mobile user, perform data encoding and decoding at server and client sites respectively using XOR-based network coding, t – time, $C_q$– valid grid cells from a query point q, $R_q$– valid range query result from a query point q, $\alpha_{q,i}$ – minimum or maximum network distance from a query point q, Q – queue, $Can_q$ – candidate's cells from a query point q, retriveIndexHeader( ) – client retrieves index header, rangeQuery(q, $UB(d_{max})$, S) – process the range query and return sub graph with candidate cells, Dijkstra(subGraph, $Can_q$, q, k) – process the query using Dijkstra's shortest path algorithm and return query result, compute a cut-off point cp using [12] %
▷% perform encoding and decoding at server and client sites respectively using adaptive XOR-network coding %
begin
1:  divide the data set S into general and specific;
2: compute a cut-off point cp;
3: query = receive_query( );
4: if query results found at client cache then
5:    return result;
6: else
7:      Listen_channel( );
8:      return result;
9: end if
10: end
1: Procedure Listen_channel( );
2: begin
3: wait t seconds for required data;
4:  if a client required data doesn't broadcast then
5:       client sends query to server;
6:       for each time slot do
7:           if empty slot exist then
8:               select specific data objects based on waiting time and priority;
9:           end if
10:      end for
11:      update cut_off point cp;
12: else if required data is received from the server then
13:          Q = 0, count = 0, $UB(d_{max})$ = 0;
14:          NCSAIHeader = retriveIndexHeader( );
15:            find out grid cells $C_q$ containing query q;
16:            read the qth row $R_q$ corresponding to $C_q$ in the matrix;
17:            sort the cells $C_i$ based on non-descending order of $\alpha_{,q,i}$ and maintained in Q;
18:            while Q is not empty do
19:                   $C_i$ = Q.pop();
20:                   count =count + $|C_i|$ ;
21:                     $UB(d_{max}) = MAX(UB(d_{max}),UB(C_q,C_i))$;
```

```
22:         end while
23:            if $count$ > = $k$ then break;
24:              ($subGraph$, $Can_q$) = rangeQuery($q$, $UB(d_{max})$, $S$);
25:              return Dijkstra($subGraph$, $Can_q$, $q$, $k$);
26:            end if
27: end if
28: end
```

Based on the above cited algorithm, it may be assumed that the client is located at node v_5 and requests k-nearest neighbor query to find "four nearest neighbor restaurants from a query point q". Then k-nearest neighbor query result is O_1, O_2, O_3 and O_4.

4.3 Continuous *k*-Nearest Neighbor Query

Continuous *k*-Nearest Neighbor (C*k*-NN) [30], [31] query widely used in many real-time applications. For example, mobile user continuously sends a query request to other mobile users while moving on the road networks called C*k*-NN query. In C*k*-NN query, once a client receives query from a mobile user, then it initially finds its located grid cell and the query path (v_1, $v_{2,...}v_L$). That's why; based on the query path the client can find data objects in each segment in road network. Each segment in the network is called as valid interval, and all valid intervals contain start/end points and these points are called as split points. Let us assume that if a client located in the same safe region requests the *k*NN query, the obtained result is the same as the partial road network is the same. So unless the client tunes into another channel and retrieves a new candidate's cell leaving the current safe region with maximum distance, the condition ($\|q, q^1\| \leq$ (UB(d_{max}) - d_k)/2)) goes unsatisfied.

Algorithm 3: Space Validation for C*k*-NN Query

Input: a query path $P(v_1,v_2,...,v_L)$
Output: valid space segments (VSS_1, VSS_2,…,VSS_L)
▷% receive_query( ) – a client waits for receiving a query from the mobile user, perform data encoding and decoding at server and client sites respectively using *XOR*-based network coding, t – time, C_q– valid grid cells from a query point q, R_q– valid range query result from a query point q, $\alpha_{q,i}$ – minimum or maximum network distance from a query point q, *start* – starting point, L – ending point, v_{start} – starting vertex, n and i – integer, G^1 – valid query path, G^{11} – query path from starting point to ending point, P – path, $|v_{start},v_i|$> ($UB(d_{max})$-d_k)/2 – if the condition is true then client need to tune into channel and retrieve new candidate cells, $VSS_n(v_{start},...,v_{i-1})$ – valid space segments from v_{start} to v_{i-1}, compute a cut-off point cp using [12] %
▷% perform encoding and decoding at server and client sites respectively using adaptive XOR-network coding %

```
begin
1:  divide the data set $S$ into general and specific;
2: compute a cut-off point $cp$;
3: query = receive_query( );
4: if query results found at client cache then
5:    return result;
6: else
7:      Listen_channel( );
8:       return result;
9: end if
10: end
1: Procedure Listen_channel();
2: begin
3: wait for $t$ seconds for required data;
4:  if the required data is not broadcast then
5:       client sends query to server;
6:       for each time slot do
7:           if empty slot exist then
8:              select specific data objects based on waiting time and priority;
9:           end if
10:        end for
11:       update cut_off point $cp$;
12: else if required data is received from the server then
13:          $start$ =1, $G^{11}$ = 0, $n$ =1;
14:          while $start$ <= $L$ do
15:                    find out the grid cells $C_q$ containing $v_{start}$ and $G^1$ = 0;
16:                    compute $UB(d_{max})$ of $v_{start}$ using Algorithm 2;
17:                       for each grid cell $C_i$ with $\alpha_{q,i}$<=$UB(d_{max})$  do
18:                             if $C_i$ belongs to $G^{11}$ then
```

```
19:                  copy content of $C_i$ to $G^1$;
20:                else
21:                   retrieve $C_i$ and add to $G^1$ and $G^{11}$;
22:               end if
23:            end for
24:            search the $k$NN of $v_{start}$ based on $G^1$
25:            find the first node $v_i$ in $P$ that $|v_{start},v_i| > (UB(d_{max})-d_k)/2$;
26:            output $VSS_n(v_{start},...,v_{i-1})$;
27:            $n$++;
28:            $start = i$;
29:       end while
30: end if
31: end
```

According to the above cited algorithmic study, it may be assumed that the client is located at node v_5 and requests continuous k-nearest neighbor query to find "continuous four nearest neighbor restaurants from a query point q. Then the ascertained continuous k-nearest neighbor query result is O_1, O_2, O_3 and O_4.

5. PERFORMANCE EVALUATIONS

Experiments have been conducted for evaluating query performance using ns-2 simulator with window 7 platform, 2.33G Intel Core 2 CPU and 3.2 GB RAM.

5.1 Experimental Process

In this simulation, real road networks data set namely Oldenburg (OL) and California (CAL) have been considered. The OL contains 6,105 nodes and 7,035 edges, and CAL contains 21,048 nodes and 21,693 edges [32] respectively. The evaluation is run on simulator which contains server, broadcast channel and clients. For simulating results, 100 clients and 500 random queries are used. In this work, we cared that 1) the size of data object is fixed at 128 bytes; 2) a set of data objects are randomly generated and uniformly distributed; 3) query issuing points are always at the network nodes; 4) network bandwidth is dynamic; 5) tuning time and access latency are measured in terms of number of bytes of data transferred in a wireless channel.

5.2 Cycle Length

Cycle length plays an important role in this simulation because it has directly impacts on the access latency. The original road network is partitioning into $2^i \times 2^i$ uniform grid cells, where i vary from 0 to 4, (i.e., the number of grids ranges from 1, to 4, to 16, to 64, and to 256). The number of grid cells in a network is set to 4^i and mapped into Hilbert curve order. The server disseminates data using (1, m) strategy by setting m to its optimal value [5**]**, we compared the performance of NCSAI with state-of-the-art NPI based on grid sizes 4^2, 4^3, and 4^4 respectively, and these are denoted as NCSAI/NPI-16, NCSAI/NPI-64 and NCSAI/NPI-256.

Table 1. Parameter Settings

Parameter	Values
k	1, 5, <u>10</u>, 15
Query scope (d/D_N)	0.01, 0.05, <u>0.1</u>, 0.2
Object density ($\|S\|/\|V\|$)	0.01, 0.05, <u>0.1</u>, 0.2
Number of cells (N)	16, <u>64</u>, 256

Table 2. Broadcast Cycle Length

Method	Index size (byte)	Data size (byte)	Cycle Length (byte)
NCSAI/NPI-16	2196	367764	389724
NCSAI/NPI-64	33300	367764	700764
NCSAI/NPI-256	526356	367764	5631324

5.3 Range Query

In a range query, a mobile client is located at query point q and issue a query with distance d then it retrieves all data objects within a d distance. In this simulation, we ignore an object density D_N but fixed at 0.1 because more number of data objects will not affects on query performance, and the radius of range query is varied from $0.01D_N$, to $0.05D_N$, to $0.1D_N$ and to $0.2D_N$. From Fig.5, we observe that with smaller radius, NCSAI-16 and NCSAI-64 have better tuning time compare to state-of-the-art NPI-16 and NPI-64 respectively. As for the access latency, NCSAI-16 has shorter access latency than NPI-16 consistently; similarly NCSAI-64 also has smaller access latency than NPI-64 in some cases because NCSAI has used both *XOR based network coding* and cache at server and client sites respectively. This set of experiments are gives a comparison between the performance of NCSAI-16 and NCSAI-64. NCSAI-16 has smaller index size, so the cycle is shorter, hence access latency is smaller. Similarly, NCSAI-64 has narrow grid cells, which provides tighter upper bound, hence smaller search space for range query, resulting in shorter tuning time. By observing range query experimental results NCSAI-64 expected to produce a better system performance compared to NCSAI-16.

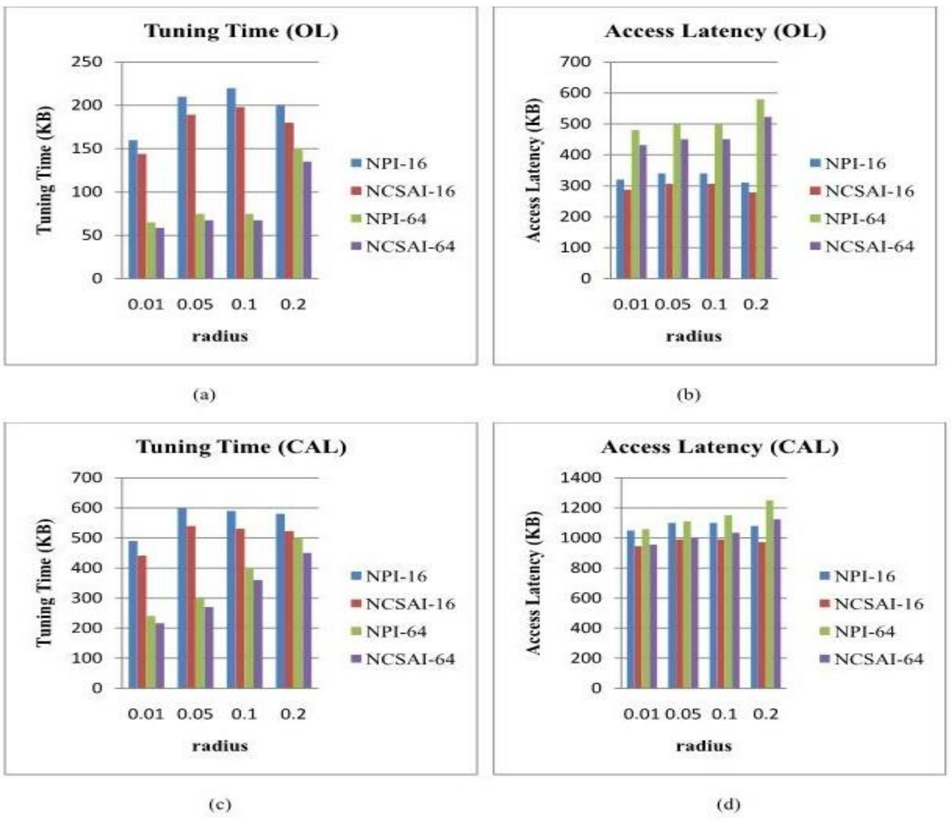
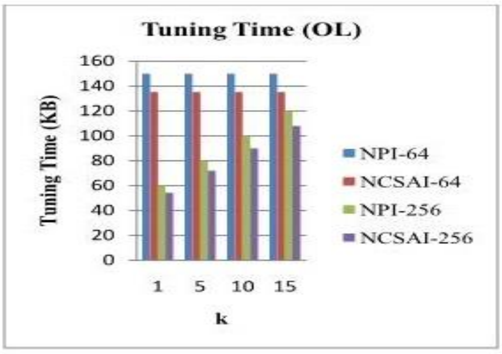


Fig.5. Performance of range queries with radius

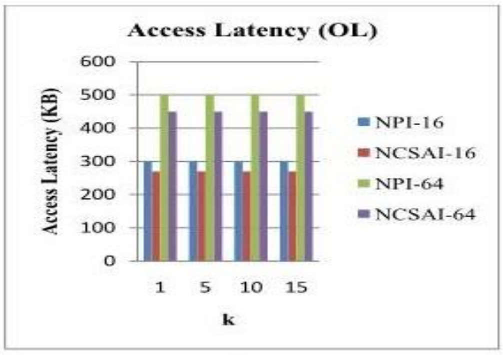
5.4 k-Nearest Neighbor Query

We evaluate the performance of state-of-the-art and proposed algorithms under various k values, as presented in Fig.11. In this experiment, objects density is fixed at 0.1 and *k* values are varied from 1, to 5, to 10, to 15. We consider only small *k* values in our experiments because mobile devices have small screen, and difficult to display a large number of NN's. We observe the performance of tuning time presented in Fig.6.(a) and Fig.6.(c), NCSAI-256 has better tuning time compared to state-of-the-art NPI-256, because it used pre-computation information carried by NCSAI and cache data at clients are enables the clients to prune the search space effectively. But, for the *k*NN queries with $k > 1$, tuning time of NCSAI increases as *k* becomes larger. Because the upper bound of the distance between the query point and k^{th} NN is enlarged by *k*, so the clients would process range query with larger radius. However, even under a relatively large *k*, the tuning time of NCSAI is still smaller compared to state-of-the-art NPI. We also observe the access latency presented in Fig.6. (b) and Fig.6.(d), NCSAI-16 has outperformed because of small index size. Similarly NCSAI-64 also has better performance compared to state-of-the-art NPI-64. However, NCSAI-256 has energy efficient, but it has large access latency because of larger index size.

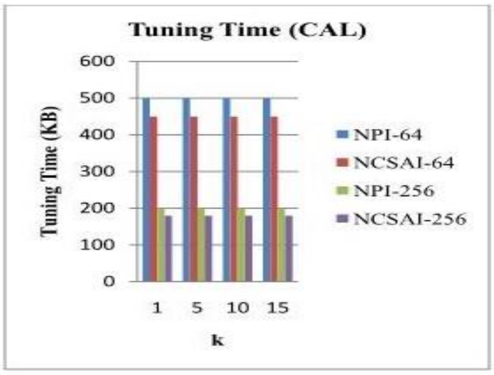
We also evaluate performance of different index with various object densities as shown in Fig.7. For simplifying the comparison, we assume that the number of grid cells is fixed at 64 for both state-of-the-art NPI and NCSAI. The performance of NCSAI-64 is more stable to the variation of the object density compared to state-of-the-art NPI-64 because NCSAI enables the clients to check the number of objects in each grid cells and compute the upper bound of the distance of the k^{th} NN. When the object density becomes dense, then the upper bound decreases because it prunes away more unnecessary grid cells.



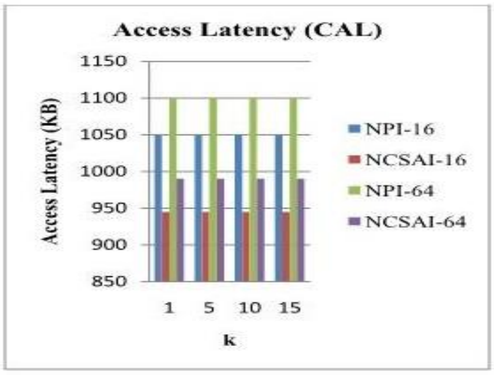
(a)



(b)

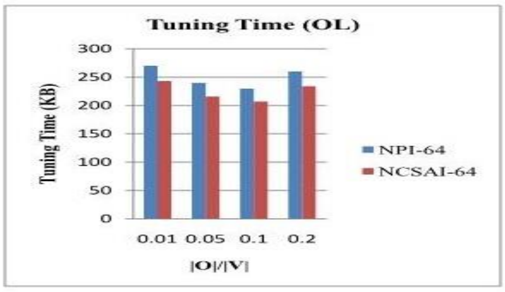


(c)

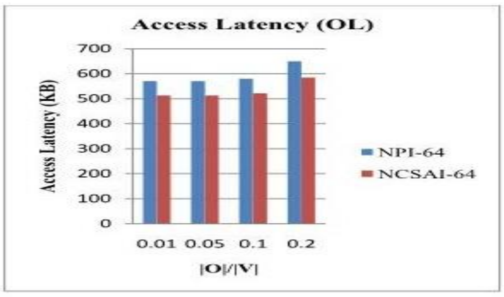


(d)

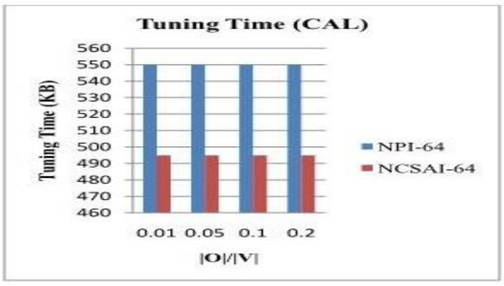
Fig.6. Performance of *k*NN queries with *k* value



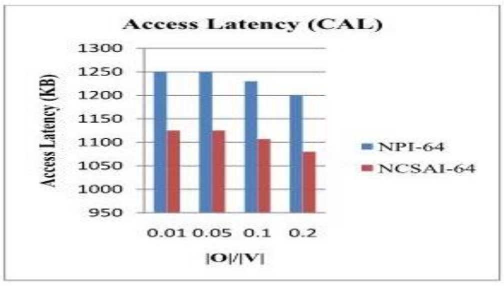
(a)



(b)



(c)



(d)

Fig.7. Performance of 10NN queries with various object densities

5 Ck-NN Query

In C*k*-NN Query, *Query Path Length Ratio* (QPLR) is defined as the ratio of the query path length to the diameter of the road network. The query path is generated by the shortest path between two randomly selected nodes in the network. In this experiment, object density is fixed at 0.1 and QPLR ranges from 0.01 to 0.2. The performance for the OL and CAL data sets for answering CNN queries with *k* = 10 as shown in Fig.8. The tuning time of NCSAI-64 varies as QPLR changes because it enables space pruning. However, tuning time of NCSAI-64 has better than state-of-the-art NPI-64. But NCSAI-64 has suffered from longer access latency because the C*k*-NN algorithm under NCSAI index, allows the clients to tune into the channel to update the partial road network and *k*NN candidates only when they arrive at the first point of a new SVS. We also evaluate the performance of state-of-the-art NPI-64 and NCSAI-64 for supporting CNN queries with different *k* values. In this experiment, we fixed QPLR at 0.1, *k* is varying from 1 to 15, and object density is fixed at 0.1 for both OL and CAL shown in Fig.9.

NCSAI-64 has more energy efficient compared to state-of-the-art NPI-64 for different *k* because the pre-computation by NCSAI helps to clients prune the needless regions and avoid the clients keeping tuning into the channel rapidly as the clients move to a new position. However, access latency of NCSAI-64 has better than state-of-the-art NPI-64 with different *k*.

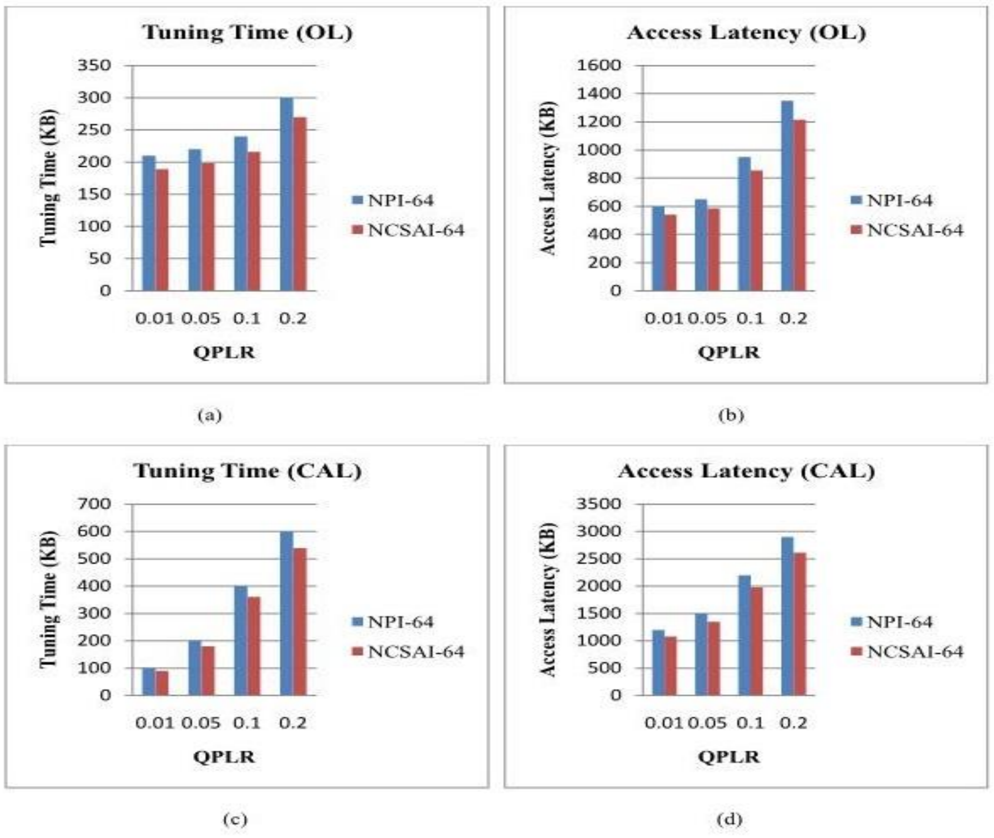


Fig.8. Performance of CNN queries with QPLR

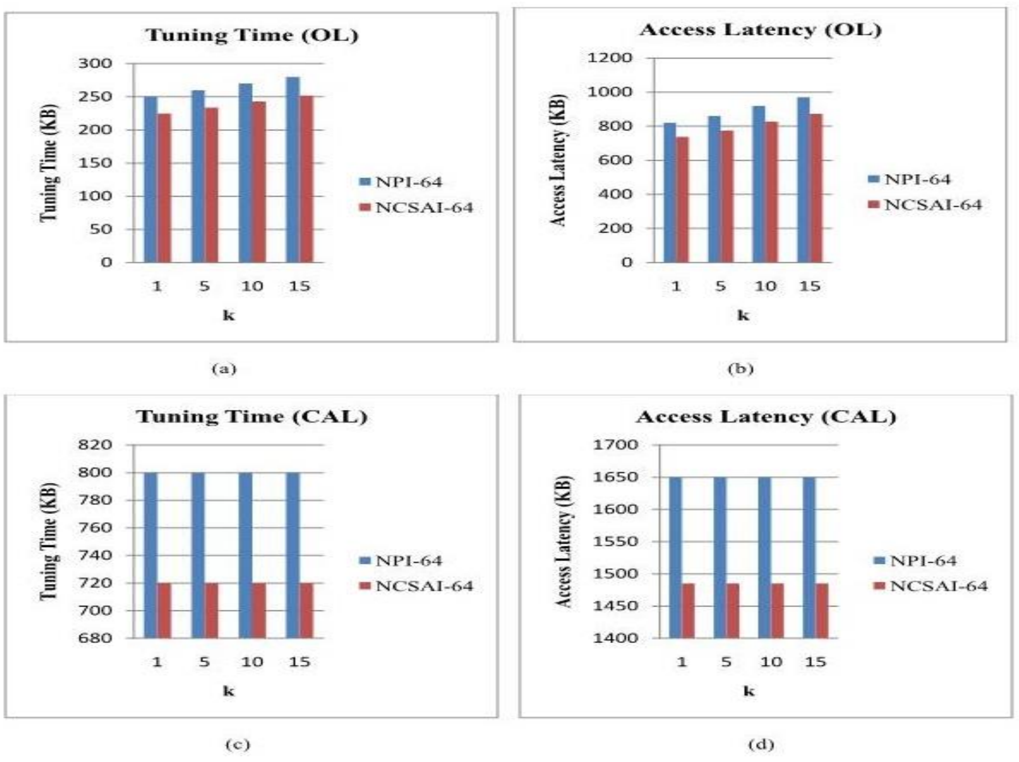


Fig.9. Performance of CNN queries with *k* values

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 04, April-2020

6. CONCLUSION

In wireless mobile environments, spatial query processing is a challenging issue due to the following limitations such as low bandwidth, limited energy and capacity, and mobility. In this work, we observed that NCSAI has improved query performance compared with state-of-the-art NPI because NCSAI effectively utilized the advantages of network coding and cached data at clients. In future, we can extent this work by incorporating an advanced network coding strategies and spatial queries.

REFERENCES

[1] D.Zhang, C.-Y. Chow, Q. Li, X. Zhang and Y. Xu, “SMashQ: Spatial Mashup Framework for k-NN queries in time-dependent Road Networks”, Distributed Parallel Data bases, vol. 31, pp. 259–287, 2013.

[2] Yu Li and Man Lung Yiu, “Route-Saver: Leveraging Route APIs for Accurate and Efficient Query Processing at Location-Based Services”, IEEE Transaction on Knowledge and Data Engineering, vol. 27, pp. 235-249, 2015.

[3] H. Kriegel, P. Kr€oger, P. Kunath, M. Renz and T. Schmidt, “Proximity queries in Large Traffic Networks”, Proceedings of 15thAnnual ACM International Symposium on Advances in Geographic Information Systems, pp. 21–28, 2007.

[4] H. Samet, J. Sankaranarayanan, and H. Alborzi, “Scalable network distance browsing in spatial databases”, Proceedings of ACM SIGMOD International Conference on Management Data, pp. 43–54, 2008.

[5] T. Imielinski, S. Viswanathan, and B.R. Badrinath, “Data on Air: Organization and Access”, IEEE Transaction on Knowledge and Data Engineering, vol. 9, pp. 353-372, 1997.

[6] Guohui Li, Quan Zhou and Jianjun Li, “A Novel Scheduling Algorithm for Supporting Periodic Queries in Broadcast Environments”, IEEE Transaction on Mobile Computing, vol.14, pp.2419- 2432, 2015.

[7] Weiwei Sun, Yongrui Qin, J. Wu, B. Zheng, Z. Zhang, P. Yu and J. Zhang, “Air Indexing for On-Demand XML Data Broadcast”, IEEE Transaction on Parallel and Distributed Systems, vol. 25, pp.1371-1381, 2014.

[8] Baihua Zheng, Wang-Chien Lee and Dik Lun Lee, “On Searching Continuous k-Nearest Neighbors in Wireless Data Broadcast Systems”, IEEE Transactions on Mobile Computing, vol.6, pp.748-761, 2007.

[9] K. Mouratidis, S. Bakiras and D. Papadias, “Continuous monitoring of spatial queries in Wireless Broadcast Environments”, IEEE Transaction on Mobile Computing, vol. 8, pp. 1297–1311, 2009.

[10] Leong Hou U, Hong Jun Zhao, Man Lung Yiu, Yuhong Li and Zhiguo Gong, “Towards Online Shortest Path Computation”, IEEE Transaction on Knowledge and Data Engineering, vol. 26, pp. 1012-1025, 2014.

[11] Weiwei Sun, Chunan Chen, Baihua Zheng, Chong Chen and Peng Liu, “An Air Index for Spatial Query Processing in Road Networks”, IEEE Transaction on Knowledge and Data Engineering, vol. 27, pp. 382-395, 2015.

[12] Sunho Kim and Sang H. Kang, “Scheduling Data Broadcast: An Efficient Cut-Off Point between Periodic and On-Demand Data”, IEEE Communications Letters, vol.14, pp.1176-1178, 2010.

[13] S. Hambrusch, C. Liu, W. Aref, and S. Prabhakar, “Query processing in broadcasted spatial index trees”, Proceedings of International Symposium on Advances in Spatial Temporal Databases, pp. 502–521, 2001.

[14] J. Xu, B. Zheng, W. Lee, and D. Lee, “Energy efficient index for querying location-dependent data in mobile broadcast environments”, Proceedings of 19th Conference on Data Engineering, pp. 239–250, 2005.

[15] B. Zheng, J. Xu, W. Lee, and L. Lee, “Grid-partition index: a hybrid method for nearest-neighbor queries in wireless location-based services”, International Journal on Very Large Data Bases, vol. 15, pp. 21–39, 2006.

[16] B. Zheng, W. Lee, and D. Lee, “Spatial queries in wireless broadcast systems”, Wireless Networks, vol. 10, pp. 723–736, 2004.

[17] B. Zheng, W. Lee, K. Lee, D. Lee, and M. Shao, “A Distributed Spatial index for Error-prone Wireless data Broadcast”, International Journal on Very Large Data Bases, vol. 18, pp. 959–986, 2009.

[18] G. Kellaris and K. Mouratidis, Shortest path computation on air indexes”, Proceedings of VLDB Endowment, vol. 3, pp. 747–757, 2010.

[19] Y. Jing, C. Chen, W. Sun, B. Zheng, L. Liu, and C. Tu, “Energy-efficient shortest path query processing on air”, Proceedings of 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 393–396, 2001.

[20] R. Ahlswede, N. Cai, S. Y. Li, and R. W. Yeung, “Network Information Flow”, IEEE Transaction on Information Theory, vol. 46, pp. 1204–1216, 2000.

[21] J. Marden and M. Effros, “The price of selfishness in network coding”, Proceedings of workshop on Network Coding Theory, 2009.

[22] Y. Sagduyu and A. Ephremides, “Cross-Layer optimization of MAC and network coding in wireless queuing tandem networks”, IEEE Transaction on Information Theory, vol. 54, pp. 554–571, 2008.

[23] H. Seferoglu, A. Markopoulou, and U. Kozat, “Network coding-aware rate control and scheduling in wireless networks”, Proceedings of IEEE International Conference on Multimedia Expo, 2009.

[24] Y. Birk and T. Kol, “Coding on demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients”, IEEE Transaction on Information Theory, vol. 52, pp. 2825–2830, 2006.

[25] E. Lubetzky and U. Stav, “Non-Linear index coding outperforming the linear optimum”, Proceedings of 48th Annual IEEE Symposium on Foundations Computer Science, 2007.

[26] M. Chaudhry and A. Sprintson, “Efficient algorithms for index coding”, Proceedings of IEEE INFOCOM Workshops, 2008.

[27] P. Sadeghi, R. Shams, and D. Traskov, “An optimal adaptive network coding scheme for minimizing decoding delay in broadcast erasure channels”, EURASIP Journal of Wireless Communication Networks, 2010.

[28] C. Chu, D. Yang, and M. Chen, “Multi-data delivery based on network coding in on-demand broadcast”, Proceedings of 9th International Conference on Mobile Data Management, 2008.

[29] Cheng Zhan, Victor C. S. Lee, Jianping Wang, and Yinlong Xu, “Coding-Based Data Broadcast Scheduling in On-Demand Broadcast”, IEEE Transaction on Wireless Communications, vol. 10, pp.3774-3783, November 2011.

[30] H.Cho and C. Chung, “An efficient and scalable approach to CNN queries in a road network”, Proceedings of 31stInternational Conference on Very Large Data Bases, pp. 865–876, 2005.

[31] M.Kolahdouzan and C. Shahabi, “Continuous *k*-nearest neighbor queries in spatial network databases”, Proceedings of Spatio-Temporal Databases Management, pp. 33–40, 2004.

[32] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S. Teng, “On trip planning queries in spatial databases”, Proceeding of 9th International Conference on Advances in Spatial Temporal Databases, pp. 923–940, 2005.

[33] Cuixiang Wang, Xing Shao , Zhi Gao , Chuanxin Zhao, and Jun Gao “Common Network Coding condition and traffic matching Supported Network Coding Aware routing for Wireless Multihop Network”,International Journal of Distributed Sensor Networks, vol. 15, pp. 1-20, 2019.

[34] Qi Wang , Xiang Zhang, Qingshan Wang , Peng Liu, and Bin Deng, “The Network Coding Algorithm Based on Rate Selection for Device-to-Device Communications”, IEEE Access, vol. 7, pp. 23396-23406, 2019,