Veeresha et al. 2017. *Int. J. Vehicle Structures & Systems*, 9(2), 87-90
ISSN: 0975-3060 (Print), 0975-3540 (Online)
doi: 10.4273/ijvss.9.2.05

International Journal of Vehicle Structures & Systems
Available online at www.maftree.org/eja

Processing of K-Nearest Neighbour Queries in Road Networks using Spatial Air Index

M. Veeresha[a] and M. Sugumaran[b]

Dept. of Computer Sci. and Engg.,
Pondicherry Engg. College, Pondicherry, India
[a]Corresponding Author, Email: veeresha.m@pec.edu
[b]Email: sugu@pec.edu

ABSTRACT:

Spatial Air Index (SAI) has been proposed for improving query performance of k-nearest neighbour queries in road networks. SAI has been effectively utilized the usage of Adaptive Cooperative Caching (ACC) and reduced search space. Experiments have been conducted for evaluated query result, the experimental result show that SAI outperform compared to state-of-the-art Network Partition Index (NPI).

KEYWORDS:

Spatial air index; K-nearest neighbour queries; Adaptive cooperative caching; Network partition index; Road networks

CITATION:

M. Veeresha and M. Sugumaran. 2017. Processing of K-Nearest Neighbour Queries in Road Networks using Spatial Air Index, *Int. J. Vehicle Structures & Systems*, 9(1), 87-90. doi:10.4273/ijvss.9.2.05.

1. Introduction

Wireless broadcast is one of the efficient strategies for disseminating data to clients in road networks and improves scalability and security. Wireless broadcast strategies have been classified into three categories and these are push-based, pull-based and hybrid scheduling strategies [1-3]. However, these strategies have been suffered from sequential data access. In Euclidian space, various scheduling strategies have been adapted for processing queries [4-5]. However, in real-time applications, query processing in road networks is essential. Air index has been adapted recently for shortest path queries in road networks. However, it doesn't address range queries and k-nearest neighbour queries and continues k-nearest neighbour queries [6]. Recently, NPI has been adapted for various spatial queries in road networks using wireless broadcast environment, but it doesn't address cache management and well scheduling strategy [7]. Motivated by this observation, SAI is proposed for spatial queries in road networks.

The original road networks may contain various data objects such as restaurants, shopping-malls, hospitals, gas-stations and schools, and these are classified into general and specific based on client's requirement, and then compute cut-off point [8]. Using grid partition strategy, original road network partitioned into small grid cells, and pre-computed general information such as diameter of each cell and minimum/maximum network distance between every pair of cells that will be carried by SAI. On server site, server broadcast general and specific data objects of SAI with network connectivity information of each cell using Hybrid Broadcast (HB) scheduling. On client site, we proposed Adaptive Cooperative Caching (ACC). In ACC, once a client receives a query request from a mobile user then it search query result in its cache, if found then it return query result. If not found, then the client send request query to region QD. If region QD doesn't contain query result then it request to nearest regions QDs in network.

If none of the region QD's doesn't respond then the client tune into channel, retrieves required information and process the query using Dijkstra's shortest path algorithm. If required data doesn't broadcast while tuning, then the client send request query to server via pull-based scheduling. While broadcasting, if any free time-slots exist then server broadcast the client requested data objects (i.e., specific data objects) along with general data objects of SAI based on longest waiting time and priority order, and updates optimal cut-off point. In this paper, SAI is proposed for k-nearest neighbour queries in road networks. K-nearest neighbour query algorithm is proposed at client site. Experiments have been conducted to compare the performance of SAI with state-of-the-art NPI.

2. Data broadcast via wireless channel using SAI

The original road networks may contain large number of data objects and these are classified into general and specific, and compute cut-off point. Using grid partition strategy, original road networks is partitioning into small grid cells and pre-computes the diameter of each cell and minimum/maximum network distance between every pair of cells, and then form NPI called SAI. At server site, server broadcast SAI with data segment on wireless channel using HB scheduling. At client site process the query based on ACC. ACC strategy consider both spatial and temporal property of data objects when making

cache replacement decisions based on client moment prediction and improves performance. In wireless mobile environment cache management play an important role and improves query performance. Cache management strategies have been considering either spatial or temporal property of data objects when making cache replacement decisions [9 - 14]. In order to improve query performance we consider both spatial and temporal property of data objects when making cache replacement decisions.

3. K-nearest neighbour query processing using SAI

K-Nearest Neighbour (k-NN) Query is defined as the client retrieves k-nearest data objects from a query point q. Like a range query, range is not fixed in k-nearest neighbour query hence it will be depend on the value of k and location of query point q. In order to find candidate cells, initially estimate the network distance d_{max} based on SAI. Once d_{max} is estimated then k-nearest neighbour query is converted into range query and retrieves candidate's data objects within a network distance d_{max} from the query point q. The d_{max} is estimated based on the upper bounds of network distance between any object in C_u and C_v and it is 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 desnoted as UB (C_u, C_v) ≤ diameter (C_u) + diameter (C_v) + $\beta_{u,v}$. Based on above lemma, d_{max} is estimated as follows. For example, if the query point q is located in cell C_q then access C_i based on non-descending order of $\alpha_{q,i}$ (i.e., C_i is visited earlier than C_j if $\alpha_{q,I} < \alpha_{q,j}$ is tie, and UB (C_q, C_i) and UB (C_q, C_j) are adapted for tie-breaker).

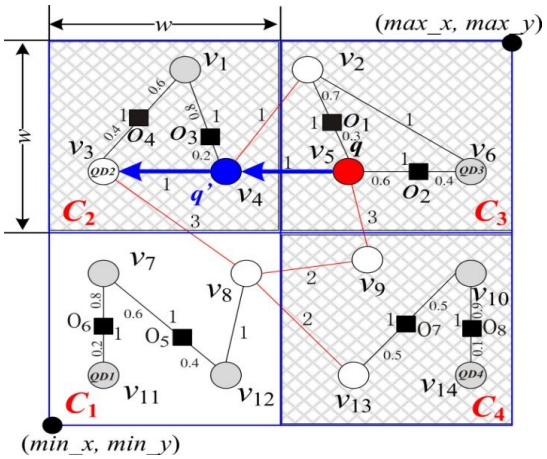

Fig. 1: Processing of k-NN query using ACC

Finally, the smaller UB value will be visited first. In this process, two parameters such count and UB (d_{max}) are used to finding k-nearest data objects. In which, count denotes the total number of data objects in all the cells visited so far, and UB (d_{max}) denotes for each visited cell C_i. For processing k-nearest neighbour query, initially compute UB (C_q, C_i) and set UB (d_{max}) to the maximum UB (C_q, C_i) found so far, and mean while check the value of $|C_i|$, and update count. This process is continuing until count reaches to value k. In this simulation, assume that the clients' locations are located at network nodes and easily extend to support cases where clients locations are locating along the network edges as shown in Fig. 1, and used Dijkstra's shortest path algorithm because it is simple and efficient for small sub-graphs.

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 mobile user, 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 – candidates cells from a query point q %*
begin
1: data set S divided into general and specific data sets;
2: compute a cut-off point cp;
3: query = receive_query( );
4: ***if*** *query results found at client/region QD cache* ***then***
5: return result;
6: ***else***
7: ***for*** *each other region QD* ***do***
8: ***if*** *query result found at any region QD's cache* ***then***
9: return result;
10: ***else***
11: Listen_channel( );
12: ***end if***
13: ***end for***
14: ***end if***
15: ***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 send query to server;
6: ***for*** *each available time slot* ***do***
7: ***if*** *empty slot exist* ***then***
8: select specific data objects based on waiting time & 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: SAIHeader = retriveIndexHeader( );
15: find out grid cells C_q containing query q;
16: read the q^{th} row R_q corresponds to C_q in 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***
27: ***end***

Assume that the client is located at node v5 and request k-nearest neighbour query to find four nearest neighbour restaurants from a query point q. Then the k - nearest neighbour query result is O1, O2, O3 and O4.

4. 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.

4.1. Experimental setup

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 [15] respectively. The evolution 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 considered 1) 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 transfer in a wireless channel.

4.2. Cycle length

In this simulation, cycle length plays an important because it is directly impacts on the access latency. The original road network is partitioning into 2i × 2i 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 4i and then mapped into Hilbert curve order. The server disseminates data using (1, m) strategy by setting optimal m value 3. In this work, the performance of SAI compared with state-of-the-art NPI based on grid sizes such as 42, 43, and 44 respectively, and these are denoted as SAI/NPI-16, SAI/NPI-64 and SAI/NPI-256. The parameter settings and broadcast cycle length are shown in Table 1 and Table 2 respectively.

Table 1: Parameter settings

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

Table 2: Broadcast cycle length

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

4.3. K-NN query

In this simulation, an object density is fixed at 0.1 and k values are varied from 1, to 5, to 10, to 15. By observing tuning time, SAI-256 has better tuning time compared to state-of-the-art NPI-256 because it used both cached data at clients and pre-computation information of SAI. For the k - NN queries with k > 1, tuning time of SAI has been increase 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 the range query with larger radius. However, even under a relatively large k, the tuning time of SAI is still smaller compared to state-of-the-art NPI. By observing access latency, SAI-16 has outperformed due to smaller index size. Similarly, SAI-64 also has better performance compared to state-of-the-art NPI-64. However, SAI-256 has energy efficient, but it suffers from large access latency due to larger index size as shown in Fig. 2. For evaluating the performance of different index with various object densities, assume that the number of grid cells is fixed at 64 for both SAI and state-of-the-art NPI because it simplifies the comparison as shown in Fig. 3.

The performance of SAI-64 has more stable to the variation of the object density compared to state-of-the-art NPI-64 because SAI enables the clients to check the number of objects in each grid cells and to calculate the upper bound of the distance of the k^{th} NN with help of cached data at clients. When the object density becomes denser, then the upper bound decreases so resulting clients reduce unnecessary grid cells.

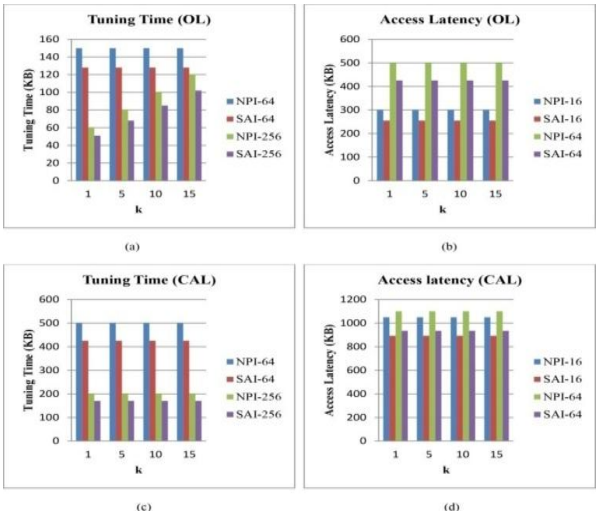

Fig. 2: Performance of k - NN queries with k value using ACC

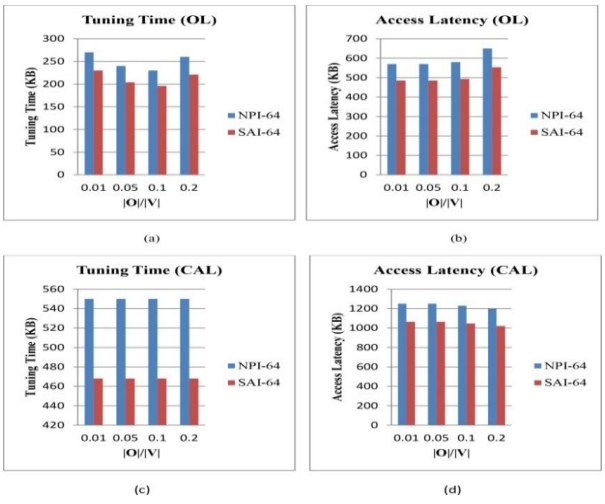

Fig. 3: Performance of 10 NN queries with various object densities using ACC

5. Conclusion

K-nearest neighbour query processing in wireless environments has been challenging issue due to limited resources such limited bandwidth, low energy and mobility. In this work, SAI has been utilized the advantage of cached data at client site and improved query performance. The experimental result has been showing that performance of SAI better than state-of-the-art NPI. In future, we can extent this work by adapting advanced caching strategies and spatial queries.

REFERENCES:

[1] G. Li, Q. Zhou and J. Li. 2015. A novel scheduling algorithm for supporting periodic queries in broadcast environments, *IEEE Trans. Mobile Computing*, 14, 419-432. https://doi.org/10.1109/TMC.2015.2398417.

[2] W. Sun, Y. Qin, J. Wu, B. Zheng, Z. Zhang, P. Yu and J. Zhang. 2014. Air indexing for on-demand XML data broadcast, *IEEE Trans. Parallel and Distributed Systems*, 25, 1371-1381. https://doi.org/10.1109/TPDS.2013.87.

[3] T. Imielinski, S. Viswanathan and B.R. Badrinath. 1997. Data on air: Organization and access, *IEEE Trans. Knowledge and Data Engg.*, 9, 353-372. https://doi.org/10.1109/69.599926.

[4] B. Zheng, W.C. Lee and D.L. Lee. 2007. On searching continuous k-nearest neighbors in wireless data broadcast systems, *IEEE Trans. Mobile Computing*, 6, 748-761. https://doi.org/10.1109/TMC.2007.1004.

[5] K. Mouratidis, S. Bakiras and D. Papadias. 2009. Continuous monitoring of spatial queries in wireless broadcast environments, *IEEE Trans. Mobile Computing*, 8, 1297-1311. https://doi.org/10.1109/TMC.2009.14.

[6] U.L. Hou, H.J. Zhao, M.L. Yiu, Y. Li and Z. Gong. 2014. Towards online shortest path computation, *IEEE Trans. Knowledge & Data Engg.*, 26, 1012-1025. https://doi.org/10.1109/TKDE.2013.176.

[7] W. Sun, C. Chen, B. Zheng, C. Chen and P. Liu. 2015. An air index for spatial query processing in road networks, *IEEE Trans. Knowledge and Data Engg.*, 27, 382-395. https://doi.org/10.1109/TKDE.2014.2330836.

[8] S. Kim and S.H. Kang. 2010. Scheduling data broadcast: An efficient cut-off point between periodic and on-demand data, *IEEE Comms. Letters*, 14, 1176-1178. https://doi.org/10.1109/LCOMM.2010.101210.101228.

[9] P.T. Joy and K.P. Jacob. 2012. A comparative study of cache replacement policies in wireless mobile networks, *Proc. Int. Symp. Adv, in Computing and Info. Tech.*, 609-619. https://doi.org/10.1007/978-3-642-31513-8_62.

[10] B. Zheng, J. Xu and D. Lee. 2002. Cache invalidation and replacement strategies for location-dependent data in mobile environments, *IEEE Trans. Computers*, 10, 1141-1153. https://doi.org/10.1109/TC.2002.1039841.

[11] W.C. Peng and M.S. Chen. 2005. Design and Performance studies of an adaptive cache retrieval scheme in a mobile computing environment, *IEEE Trans. Mobile Computing*, 4, 29-40. https://doi.org/10.1109/TMC.2005.9.

[12] W.C. Peng and M.S. Chen. 2005. Shared data allocation in a mobile computing system: Exploring local and global optimization, *IEEE Trans. Parallel & Distributed Systems*, 16, 374-384. https://doi.org/10.1109/TPDS.2005.50.

[13] L. Yin and G. Cao. 2006. Supporting cooperative caching in adhoc networks. *IEEE Trans. Mobile Computing*, 5, 77-89. https://doi.org/10.1109/TMC.2006.15.

[14] Q. Zhu, D.L. Lee and W.C. Lee. 2011. Collaborative caching for spatial queries in mobile P2P networks, *IEEE Int. Conf. Data Engg.*, 279-290.

[15] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios and S. Teng. 2005. On trip planning queries in spatial databases, *Proc. 9th Int. Conf. Adv. Spatial Temporal Databases*, 923-923. https://doi.org/10.1007/11535331_16.