HOC-Tree: A Novel Index for Efficient Spatio-Temporal Range Search

Jun Long[1,2], Lei Zhu[1,2], Chengyuan Zhang[1,2(✉)], Shuangqiao Lin[1,2], Zhan Yang[1,2], and Xinpan Yuan[3]

[1] School of Information Science and Engineering, Central South University, Changsha, People's Republic of China
{jlong,leizhu,cyzhang,linshq,zyang22}@csu.edu.cn
[2] Big Data and Knowledge Engineering Institute, Central South University, Changsha, People's Republic of China
[3] School of Computer, Hunan University of Technology, Zhuzhou, China
xpyuan@hut.edu.cn

Abstract. With the rapid development of mobile computing and Web services, a huge amount of data with spatial and temporal information have been collected everyday by smart mobile terminals, in which an object is described by its spatial information and temporal information. Motivated by the significance of spatio-temporal range search and the lack of efficient search algorithm, in this paper, we study the problem of spatio-temporal range search (STRS), a novel index structure is proposed, called HOC-Tree, which is based on Hilbert curve and OC-Tree, and takes both spatial and temporal information into consideration. Based on HOC-Tree, we develop an efficient algorithm to solve the problem of spatio-temporal range search. Comprehensive experiments on real and synthetic data demonstrate that our method is more efficient than the state-of-the-art technique.

Keywords: Hilbert curve · Spatio-temporal · Range search
HOC-Tree

1 Introduction

With the rapid development of mobile computing and Web services, a huge amount of data [8,9,11,15] with spatial and temporal information have been collected everyday by smart mobile terminals, such as smart phones, tablets, wearable devices etc., or devices of Iot which are equiped with GPS or wireless modules. In addition, Location Based Services (LBS) and social network services provide users with location-dependent information and services in dailylife. Everyday a vast number of pictures [10,17,19] and texts with geotags [21–23]and timestamps are posted to Fackbook or Instagram. Foursquare supports more than 45 million users who have checked-in more than 5 billion times at over 1.6 million businesses. Users can search any interested information by specified time interval and geolocation.

In this paper, we study an important search problem in spatio-temporal data query area, named spatio-temporal range search (STRS for short). Spatio-temporal range search aims to retrieval all spatio-temporal objects whose location is within a specific geographical region during a time region. In many application scenarios, it plays an important role for data management and geo-social networks. For example, in location-based social networks platforms, such as Facebook, Twitter, Weibo, etc. Users prefer to make friends with the people who usually do daily activities in the same geographic region and same time range, because same daily activities like shopping, doing outdoor exercises, going to cinema, etc. are important factors to establish relationships. Thus according to the posts with spatio-temporal data, they can find the users who have the same hobbies within a given area and given time interval, shown in Fig. 1. The red square is the geographical range of search for daily activities. Likewise, location based services like Facebook's Nearby and Foursquare's Radar return the friends that recently checked-in at close proximity to a user's current location [1,8,14,16]. In the big data age, as swift growth of the amount of spatio-temporal data, spatio-temporal range search has become a hot issue in data searching and management area.

Fig. 1. An example of spatio-temporal search in ocation-based social network services

Motivation. The challenges for the problem of spatio-temporal range search are two-fold. (i) Due to the massive amount of spatio-temporal objects in lots

of important applications, large-scale heterogeneous social networks with spatial and temporal information have been constructed. How to efficient management and access geo-social data is a core problem. (ii) For the various application requirements in social network services, high efficient search algorithms need to be developed by combining spatial and temporal features of social data.

Motivated by the significance of spatio-temporal range search and the lack of efficient search algorithm, we propose a novel spatio-temporal index structure, named HOC-Tree based on Hilbert curve and OC-Tree. Besides, we develop an efficient range search algorithm for STRS problem. OC-Tree is an important index structure in spatial database area. It is most often used to partition a three-dimensional space by recursively subdividing it into eight octants. HOC-Tree is a nature extension of OC-Tree, but it not only inherits the valuable properties in 3-dimensional partition, such as the data that close in space and close in time are partition into same cell, but also provides an efficient 3D Morton code generation mechanism, which can easily and effectively combine the spatial and temporal information together to support spatio-temporal search.

Contributions. To summarize, our key contributions in this paper are summarized as follows:

(1) We propose a novel spatio-temporal index based on Hilbert curve and OC-Tree named HOC-Tree to solve the problem of spatio-temporal range search. To the best of our knowledge, this study is the first time to design a novel spatio-temporal indexing mechanism for efficient spatio-temporal range search.
(2) We develop an efficient spatio-temporal range search algorithm based on HOC-Tree.
(3) We conduct comprehensive experiments on real and synthetic datasets. The results show that our method can solve spatio-temporal range search effectively and efficiently, and it outperforms the state-of-the-art approaches.

Roadmap. The rest of the paper is organized as follows: We present the related work in Sect. 2. Section 3 formally defines the problem and describes the index structure. We elaborate the search algorithm in Sect. 4 and extensive experiments are presented in Sect. 5. Finally, we offer conclusions in Sect. 6.

2   Related Work

In this section, we review geo-social networks queries and collective spatial queries, which are two kinds of techniques related to our works.

Geo-Social Networks Queries. A typical geo-social network [9,13,18] combines social networks techniques [12,16] and spatial data queries techniques. Many research findings from academia and techniques applying to industry have

been proposed. In industrial circle, the most famous social networks platform, Facebook, provided a location-based social network service named *Nearby* [1] which aims to find the friends who are in the neighborhood of a user currently. Geoloqi is another analogous platform for building location aware applications. It provides the service which notifies users when their friends get into a certain geographical region. Uber is a advanced mobile Internet platform for texi service based on geo-location information of texi drivers and riders. The riders can search the near drivers around them and send messages for request. These applications just only focus on the spatial attributes of data on social networks or cloud platforms for range search. In academic circles, the problem of spatio-temporal search is concerned by lots of researchers. In [1], armenatzoglou et al. proposed a general framework that offers flexible data management and algorithmic design. The nearest star group query contained in the framework returns the k nearest subgraphs of m users. In [4], Liu et al. proposed propose the k-Geo-Social Circle of Friend Query which aims to finds the group g of $k + 1$ users, which (i) is connected, (ii) contains u, and (iii) minimizes the maximum distance between any two of its members. In [6], Scellato et al. proposed three more geo-social networks metrics: (i) average distance (ii) distance strength, and (iii) average triangle length. In [20], Yang et al. developed a hybrid index named Social R-Tree to solve the problem of socio-spatial group query. The studies mentioned above did not combine the spatial and temporal attributes of objects in database for searching.

Geo-Social Networks Queries. Collective spatial query is another important problem. In [24], Zhang et al. presented a novel spatial keyword query problem called the m-closest keywords (mCK) query, which aims to aims to find the spatially closest tuples which match m user-specified keywords. They proposed a new index named bR*-tree extended from the R*-tree to address this problem. The R*-tree designed by Beckmann et al., which incorporates a combined optimization of area, margin and overlap of each enclosing rectangle in the directory [7]. In [3], Guo et al. proved that answering mCK query is NP-hard and designed two approximation algorithms called *SKEC*a and *SKEC*a+. In [2], Deng et al. proposed a generic version of closest keywords search named best keyword cover which considers inter-objects distance as well as the keyword rating of objects. In [5], Long et al. studied two types of the CoSKQ problems, MaxSum-CoSKQ and Dia-CoSKQ. These studies aim to solve the problem of spatial keyword queries to find a set of objects. They did not develop efficient index structure and search algorithms for range search in a specific geographical area and time interval.

3   Model and Structure

This section first presents a Definition of problem, then describes the proposed data structure, named HOC-Tree, which based on Hilbert curve and OC-Tree. Table 1 below summarizes the symbols used frequently throughout the paper.

Table 1. The summary of notations

Notation	Definition
D	A given data set of spatio-temporal data
o	A spatio-temporal object
x_i	The longitude of a spatio-temporal data
y_i	The latitude of a spatio-temporal data
t_i	The timestamp of a spatio-temporal data
h_n	The value of Morton order in Hilbert curve
L	The deepest level of HOC-Tree
ψ	The division threshold value for a node in HOC-Tree
v	The Morton order of a leaf node in HOC-Tree
q	A spatio-temporal range search ($[x_{min}, x_{max}], [y_{min}, y_{max}], [t_{start}, t_{end}]$)
d	The Euclidean distance between two points in spatial space

3.1   Problem Definition

Definition 1 (Spatio-temporal Object Set). *A spatio-temporal object set can be defined as $D = \{o_1, o_2, \ldots, o_n\}$. Each spatio-temporal object o is associated with a spatial location $o.(x_i, y_i)$ and the timestamp $o.t_i$.*

Definition 2 (Spatio-temporal Range Search (STRS)). *Given a spatio-temporal objects data set D, a range query is defined as $q([x_{min}, x_{max}], [y_{min}, y_{max}], [t_{start}, t_{end}])$ where $([x_{min}, x_{max}], [y_{min}, y_{max}])$ is the query spatial region and $([t_{start}, t_{end}])$ is the query temporal interval, this work aims to select all the records which satisfy the query q from D.*

3.2   Index Structure

In this section, we introduce a novel spatio-temoral index, named HOC-Tree, which is based on OC-Tree and Hilbert curve. This data structure is the key technique of this work.

As it will be shown in Subsect. 4.1, the more subspaces overlapping with range query q, the more time will be consumed when searching HOC-Tree. To solve this problem, a *MBRsign* tag data structure is used to reduce non-promising nodes access, which can avoid unnecessary I/O costs. For each subspace, the spatial locations of all the points in it can be associated with a minimum bounding rectangle (MBR), so a *MBRsign* tag is maintained for each non-empty leaf node to keep the MBR information. For a given range query q, the covering non-empty leaf nodes which don't satisfy the spatial constraint will not be accessed in searching process with the help of tags. HOC-Tree keeps two end points information of the MBR, which only require 16 bytes for each non-empty leaf node. The more detail of using the tags will be described particularly in Subsect. 4.1, where elucidates the search algorithms. Figure 2 illustrates the structure of HOC-Tree with *MBRsign* tags.
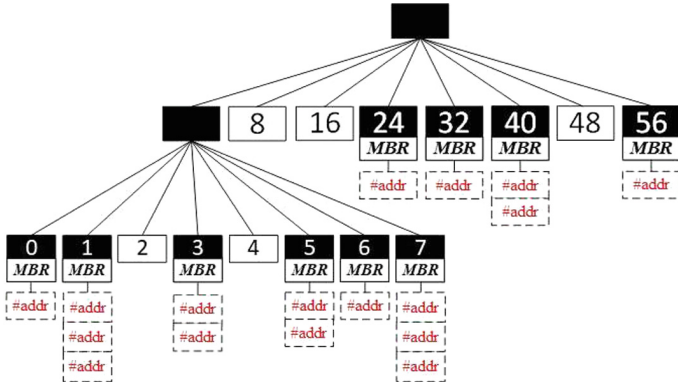
Fig. 2. An example of HOC-Tree with *MBRsign* tags

The black blocks represent non-empty nodes which contains a list of spatio-temporal data locations while the white blocks represent empty nodes. Each leaf node is labeled by its Morton order value according to our approach as mentioned above, and tags are kept for them to maintain the MBR.

4   Spatio-Temporal Query Algorithms

This section gives exhaustive description of spatio-temporal range search based on HOC-Tree.

4.1   Range Search Algorithm

Range query q is an essential function in spatio-temporal data processing. In our algorithms as shown in Algorithm 1, this work is done in several stages. The input query $q = ([x_{min}, x_{max}], [y_{min}, y_{max}], [t_{start}, t_{end}])$ is in three-dimensional space, where $[x_{min}, x_{max}]$ gives the range of longitude, $[y_{min}, y_{max}]$ gives the range of latitude and $[t_{start}, t_{end}]$ gives the time interval. The output S is a set of entries inside spatio-temporal query q. This algorithm only accesses the optimized nodes when searching HOC-Tree. A prune process is executed to check the entries whether they satisfy the query range or not and remove false positives to refine results.

Mapping Hilbert Curve Values: For a given range query q, the Hilbert curve values of covering spatial spaces can be calculated immediately according to the region $([x_{min}, x_{max}], [y_{min}, y_{max}])$ of q. The function *getHilbertValues()* maps the rectangle region into a set of one-dimensional values in line 2.

Finding Spatio-Temporal Covering Cubes: Before searching HOC-Tree in corresponding regions locally, the function *getOverlappingCubes()* line in 6 computes the covering nodes which overlap with three-dimensional query range.

Algorithm 1. Spatio-Temporal Range Search

Input: $q : x_{min}, x_{max}, y_{min}, y_{max}, t_{start}, t_{end}$.
Output: S :set of entries inside query range q.
1: $S \leftarrow \emptyset; CoveringSpacialSpaces \leftarrow \emptyset; RegionSet \leftarrow \emptyset;$
2: $CoveringSpacialSpaces \leftarrow getHilbertValues(x_{min}, x_{max}, y_{min}, y_{max});$
3: $RegionSet \leftarrow getRegions(CoveringSpacialSpaces);$
4: $CoveringNodes \leftarrow \emptyset; Q.L^f \leftarrow \emptyset; Q.L^p \leftarrow \emptyset;$
5: **for** each region $\in RegionSet$ **do**
6:     $CoveringNodes \leftarrow getOverlappingCubes(x_{min}, x_{max}, y_{min}, y_{max}, t_{start}, t_{end});$
7:     $N^f, N^p \leftarrow Identify(CoveringNodes);$
8:     **for** each Node $v \in N^p$ **do**
9:         **MBRCheck**$(MBRsign_v, x_{min}, x_{max}, y_{min}, y_{max});$
10:     **end for**
11:     **for** each Node $v \in N^p$ *that survive from* **MBRCheck** *process* **do**
12:         $Q.L^p \leftarrow getEntriesList(v);$
13:     **end for**
14:     **for** each Node $v \in N^f$ **do**
15:         $Q.L^f \leftarrow getEntriesList(v);$
16:     **end for**
17:     $S \leftarrow Q.L^f + Prune(Q.L^p);$
18: **end for**
19: **return** S.

The covering cubes can be partial or full. The left part of Fig. 3 shows a spatio-temporal range query q (the shaded cube) which would overlap multiple sub-spaces. For simplicity, the partition of space does not present here. The cubes overlapping with query range in spatial dimension is illustrated in the right part of Fig. 3, where the deepest level L of the HOC-Tree is 4. We can see that cube A has a full spatial overlap while the rest cubes have partial spatial overlap.

For each covering node, it needs to be searched HOC-Tree to get the list of addresses refer to the locations of data point. All the points in fully spatio-temporal overlapping cubes will satisfy the spatio-temporal range search which do not need to do an additional refinement step. **Algorithm 2 Identify** distinguishes these two kinds of covering nodes by N^f and N^p, where N^f denotes the set of fully covering nodes and N^p denotes the set of partially covering nodes. The identification of full overlaps helps to reduce the computation time, which can avoid unnecessary CPU checking overhead in the refinement step.

Confirming Non-empty Covering Nodes: The benefit of coupling the spatial and temporal information in our index will be more clear in this stage. As shown in Fig. 3 (right part), the overlap of cube B with query's spatial dimensional area is very small w.r.t cube A, which has a full overlap. If searching the index without any spatial discrimination, then a very small overlap (i.e., the cube B) will need the same I/O costs with that of a full overlap (i.e., the cube A). As a result, many false positive results will be collected, which have to be later pruned through the spatial criteria. Especially when the data is skew, there might be a lot of empty partial covering nodes. This case can happen because the points in
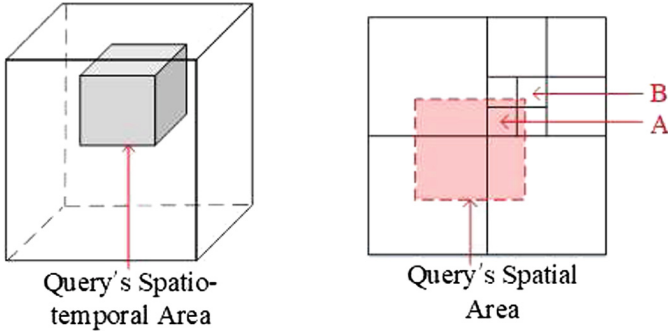
Fig. 3. Spatio-temporal range search

Algorithm 2. Identify(CoveringNodes)

Input: *CoveringNodes.*
Output: N^p, N^f
 1: $N^f \leftarrow \emptyset; N^p \leftarrow \emptyset;$
 2: **for** each region $\in CoveringNodes$ **do**
 3:     **if** v is a full overlapping leaf node **then**
 4:         add v into N^f;
 5:     **end if**
 6:     **if** v is a partial overlapping leaf node **then**
 7:         add v into N^p;
 8:     **end if**
 9: **end for**
10: **return** N^p, N^f.

---

that cube do not satisfy with the spatial criteria of q. In line 8 to 10, the information kept in *MBRsign* tag is used to check whether the MBR overlap with the spatial criteria or not. The checking is just needed in partial covering nodes because the points in full overlaps will all satisfy the spatio-temporal criteria. The confirmation of non-empty spatial covering nodes can efficiently reduce the number of false positive results in region search. As shown in Fig. 4, the Morton values of overlapping nodes (i.e. the nodes in the rectangle marked with dotted lines) is obtained by the function *getOverlappingCubes()*, which overlap with the range query given in Fig. 3. For simplicity, the further division of the Node v_3 as shown is omitted here. Then with the help of *MBRsign* tag, it can further confirm the non-empty covering nodes (i.e. the Node v_3, the Node v_7 and the Node v_{24}), which need to be searched in HOC-Tree.

Searching HOC-Tree: Spatio-temporal adjacent nodes will be stored nearby each other by this encoding in HOC-Tree. Identifying full and partial covering nodes helps to reduce the computation time, while confirming non-empty spatial covering nodes can reduce the number of I/O during searching HOC-Tree. Furthermore, the property of Hilbert curve can ensure that the generated Morton

value of query range will contain all the valid points, which discussed in Subsect. 3.2. According to the stages described above, the algorithm can get a set of full covering nodes N^f and a set of non-empty partial covering nodes N^p which need to be searched in HOC-Tree. Line 11 to 16 give the search result by $Q.L^f$ and $Q.L^p$, where the notation $Q.L^f$ and $Q.L^p$ denote the sets of entries in full and partial covering nodes respectively.
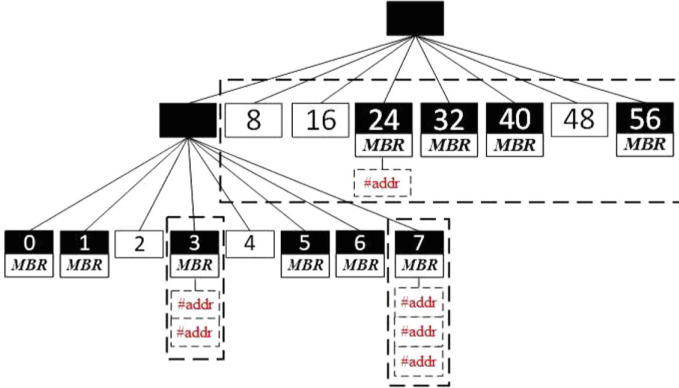

Fig. 4. Query overlapping nodes and non-empty covering nodes

Refining Results: The entries in $Q.L^p$ which have partial overlap need further refinement. There might be some points in partial overlapping nodes that not satisfy with the spatio-temporal query range. **Algorithm 3 Prune** checks each entry in $Q.L^p$ whether it is inside query range or not and removes unrelated results immediately.

Algorithm 3. Prune($Q.L^p$)

Input: $Q.L^p$
Output: $results$
 1: $Results \leftarrow \emptyset$;
 2: **while** $Q.L^p \neq \emptyset$ **do**
 3:   **for** each entry $e \in Q.L^p$ **do**
 4:     **if** query range contains e **then**
 5:       add e into results;
 6:     **end if**
 7:   **end for**
 8: **end while**
 9: **return** $results$.

---

5   Experiments

5.1   Experiment Settings and Datasets

With the implementation of HOC-Tree, a comprehensive experimental evaluation is conducted to verify the performance of the scheme in a real cloud environment. The locations of all datasets were scaled to the two-dimensional space $[0, 10000]^2$, and the timestamp of all datasets were scaled to $[0, 5000]$. In addition, the spatial region grew from $200 * 200$ to $1000 * 1000$, the time interval varied from 200 to 1000, and k changed from 10 to 500. By default, spatial region, time interval and k were set to **600**, **600**, **100** respectively. We conducted experiments on a 3 GHz Intel Core i5 2320 CPU and 8 GB RAM running 64-bit Ubuntu 16.04 LTS.

Three different datasets were used in the experiments, one of which was a synthetic uniform dataset (UN) generated by program, and others were real-world datasets, described as following: the first one was collected in Geolife project [22] (GL) by 182 users from April 2007 to August 2012, the second one was T-Drive [23, 24] (TD) generated by 33 thousand taxis on Beijing road network over a period of 3 months.

For accurate analyzing and evaluating, STEHIX was chosen as comparative object, which has a similar index scheme with ours. In each experimental case, the process was repeated for 5 times and the average value was reported. For the HOC-Tree in all the tests, the deepest level L was set to 16 and the division threshold value ψ was set to 200.

5.2   Performance Evaluation

Evaluation on Different Datasets. A series of evaluation was performed on index construction time, index size and data query performance separately against three datasets GL, TD and UN, where other parameters had default settings.
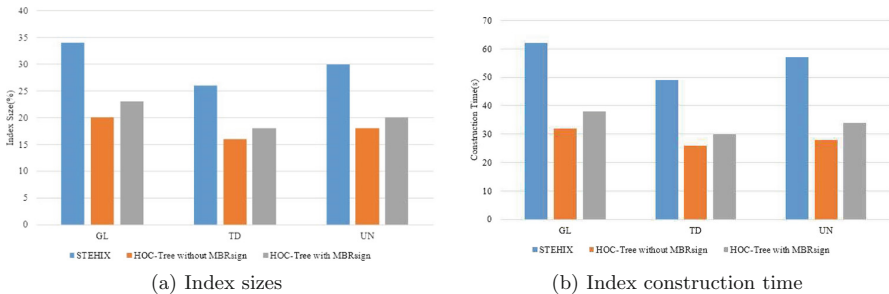

(a) Index sizes   (b) Index construction time

Fig. 5. Evaluation on different datasets

Figure 5(a) depicts the rate of space occupying by the index sizes. STEHIX requires more space due to the two kinds of indices (called *s-index* and *t-index*)

kept for all the entries. The storage cost of the STEHIX increases faster in larger datasets. In contrast, an index record is maintained for each entry only once so that our index saves more space in memory. Particularly, HOC-Tree with *MBRsign* tag occupies a very small index size compared with HOC-Tree without *MBRsign* tag. Figure 5(b) shows the difference of construction time between HOC-Tree and STEHIX. Due to the simple split and code algorithms of an HOC-Tree, our method has a shorter constructing time as compared to STEHIX, which need to traverse two indices during the construction.

The experiment results of range query on different datasets are shown in Fig. 6, where the spatial region and time interval were both set 600. The query performance was measured by computing the duration time between when the regions started searching and when client received all accurate results.
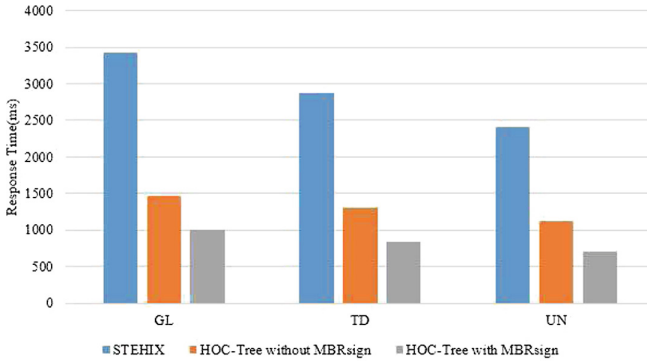

Fig. 6. Data query performance on various datasets

The HOC-Tree demonstrates superior performance in comparison with STE-HIX. Our analysis is as follows, STEHIX calculates the number of addresses in *s-index* and *t-index* separately to choose the high-selectivity list for further retrieval. In sense that, each query will decompose into two processes to collect results in temporal dimension and spatial dimension, which will provoke more I/O costs. On the other hand, STEHIX uses a period time T to divide all entries in temporal dimension because of the periodicity in timestamps, which means if let $T = 24$ (a period of 24 h is a cycle) and divide T into several segments such as 8 segments, then all the entries will map into the 8 segments by their temporal information. For a given temporal range query, all results returned by STEHIX are confused by modulo value, which have same time intervals but different in dates. Therefore, it will take more time to remove false positive results, which delays the response time in queries.

HOC-Tree with *MBRsign* tag improves a little efficiency comparing with HOC-Tree without *MBRsign* tag for uniform data (UN), because the benefit of MBR information maintained in *MBRsign* is apparently in skew data such as GL and TD.

Evaluation on the Effect of Varies Extent in STRS. A series experiments was conducted to investigate the effect of spatial region and temporal interval respectively. These experiments purposed to present the benefits of coupling spatio-temporal information in HOC-Tree and maintaining the *MBRsign* tag.

In order to show the trend of performance change in different spatial region, the temporal interval was set as the default value 600. Two representative datasets were used in the experiments, one was real-world dataset (TD) and another was uniform synthetic dataset (UN). As shown in Fig. 7(a) and (b), the time cost on different datasets is plotted.
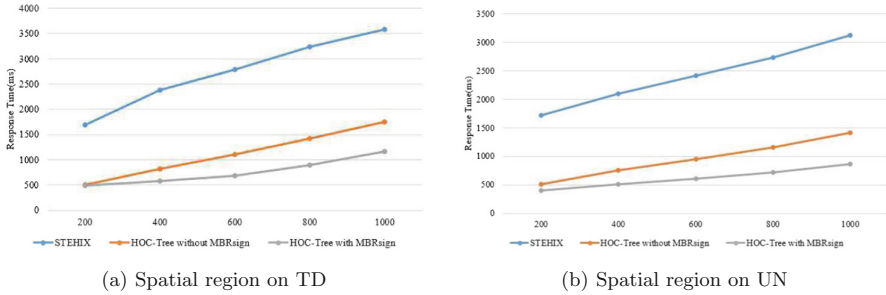

(a) Spatial region on TD   (b) Spatial region on UN

Fig. 7. The effect of varies extent in STRS

Apparently, larger spatial region means larger spatial search area, which results in longer response time. Therefore, both of two indexes perform better when the spatial region is small. On the other hand, the performance of range query in uniform dataset is better than real-world dataset, which mainly because that real-world dataset is a skew data. As is evident from the experiments, HOC-Tree shows an improvement over STEHIX especially when the spatial region is large. Larger spatial query range leads to much more unrelated entries identified as candidates in STEHIX. It spends much more running time and CPU cost because of the high computation for refinement step. However, our index performs better due to the non-fully-decoupling spatial and temporal properties so that all the points are placed by their spatio-temporal proximity in HOC-Tree which can help to reduce I/O load when searching trees. For a given three-dimensional query, HOC-Tree can immediately locate the covering nodes and explore the corresponding HOC-Tree which is owing to the efficient nodes' pruning and the use of Morton value. As pointed out earlier, STRS identifies as more full covering nodes as possible during executing query operation, which helps to reduce the CPU cost for checking fully satisfied entries.

To evaluate the effect of temporal interval on response time of HOC-Tree and STEHIX, experiments were conducted in the same manner with the previous one and spatial region was set as the default value 600. The experimental results are demonstrated in Fig. 8(a) and (b).
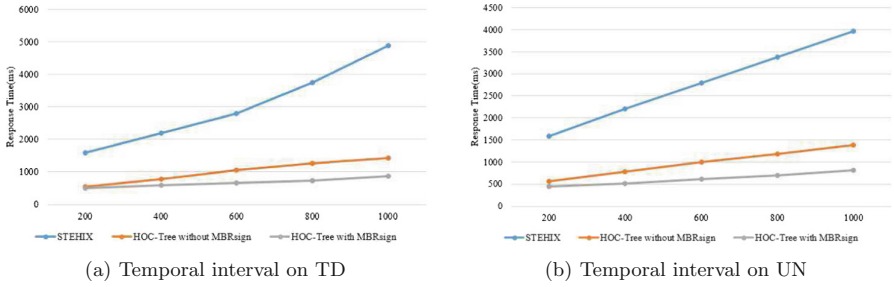
(a) Temporal interval on TD   (b) Temporal interval on UN

Fig. 8. Performance effected by temporal interval

A large temporal query range would cover a lot of partial overlapping nodes, which fully satisfy with temporal restriction but non-fully satisfy with spatial restriction. As the temporal query range becomes larger in STEHIX, all these nodes have to be accessed by *s-index* and *t-index*, which increases the number of I/Os obviously and there are much more candidates to check in the refinement step while HOC-Tree has removed a lot earlier. Because, a *MBRsign* tag data structure is designed in HOC-Tree reduce non-promising nodes access so that these nodes can be removed earlier to avoid unnecessary I/O costs. Figure 8(a) and (b) demonstrate the running time of HOC-Tree with and without the tag, and our index performs better especially for skewed data. In such a scenario, the *MBRsign* tag makes full use of non-fully-decoupled spatial and temporal information to confirm non-empty spatial covering nodes and thus many unnecessary I/O load can be avoided. For uniform dataset (UN), tag is still helpful when there are large number of empty partial overlapping nodes.

6   Conclusions

The problem of spatio-temporal search is very significant due to the increasing amount of spatio-temporal data collected in widely applications. The proposed HOC-Tree is based on Hilbert curve and OC-Tree. Based on HOC-Tree, we design an efficient algorithm to solve the problem of spatio-temporal range search. The results of our experiments on real and synthetic data demonstrate that HOC-Tree is able to achieve a reduction of the processing time by 60–80% compared with prior state-of-the-art methods.

References

1. Armenatzoglou, N., Papadopoulos, S., Papadias, D.: A general framework for geo-social query processing. PVLDB **6**(10), 913–924 (2013)
2. Deng, K., Li, X., Lu, J., Zhou, X.: Best keyword cover search. IEEE Trans. Knowl. Data Eng. **27**(1), 61–73 (2015)
3. Guo, T., Cao, X., Cong, G.: Efficient algorithms for answering the m-closest keywords query. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, 31 May–4 June 2015, pp. 405–418 (2015)
4. Liu, W., Sun, W., Chen, C., Huang, Y., Jing, Y., Chen, K.: Circle of friend query in geo-social networks. In: Lee, S., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012. LNCS, vol. 7239, pp. 126–137. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29035-0_9
5. Long, C., Wong, R.C., Wang, K., Fu, A.W.: Collective spatial keyword queries: a distance owner-driven approach. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, 22–27 June 2013, pp. 689–700 (2013)
6. Scellato, S., Noulas, A., Lambiotte, R., Mascolo, C.: Socio-spatial properties of online location-based social networks. In: Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, 17–21 July 2011 (2011)
7. Beckmann, N., Kriegel, H.P., Schneider, R., et al.: The R*-tree: an efficient and robust access method for points and rectangles. ACM SIGMOD Rec. **19**(2), 322–331 (1990)
8. Wang, Y., Lin, X., Wu, L., Zhang, W.: Effective multi-query expansions: collaborative deep networks for robust landmark retrieval. IEEE Trans. Image Process. **26**(3), 1393–1404 (2017)
9. Wang, Y., Lin, X., Wu, L., Zhang, W., Zhang, Q.: Exploiting correlation consensus: towards subspace clustering for multi-modal data. In: Proceedings of the ACM International Conference on Multimedia, MM 2014, Orlando, FL, USA, 03–07 November 2014, pp. 981–984 (2014)
10. Wang, Y., Lin, X., Wu, L., Zhang, W., Zhang, Q.: LBMCH: learning bridging mapping for cross-modal hashing. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, 9–13 August 2015, pp. 999–1002 (2015)
11. Wang, Y., Lin, X., Wu, L., Zhang, W., Zhang, Q., Huang, X.: Robust subspace clustering for multi-view data by exploiting correlation consensus. IEEE Trans. Image Process. **24**(11), 3939–3949 (2015)
12. Wang, Y., Lin, X., Zhang, Q., Wu, L.: Shifting hypergraphs by probabilistic voting. In: Tseng, V.S., Ho, T.B., Zhou, Z.-H., Chen, A.L.P., Kao, H.-Y. (eds.) PAKDD 2014. LNCS (LNAI), vol. 8444, pp. 234–246. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06605-9_20
13. Wang, Y., Wu, L.: Beyond low-rank representations: orthogonal clustering basis reconstruction with optimized graph structure for multi-view spectral clustering. Neural Netw. **103**, 1–8 (2018)
14. Wang, Y., Wu, L., Lin, X., Gao, J.: Multiview spectral clustering via structured low-rank matrix factorization. IEEE Trans. Neural Netw. Learn. Syst. **29**, 4833–4843 (2018)

15. Wang, Y., Zhang, W., Wu, L., Lin, X., Zhao, X.: Unsupervised metric fusion over multiview data by graph random walk-based cross-view diffusion. IEEE Trans. Neural Netw. Learn. Syst. **28**(1), 57–70 (2017)
16. Wu, L., Wang, Y., Gao, J., Li, X.: Deep adaptive feature embedding with local sample distributions for person re-identification. Pattern Recognit. **73**, 275–288 (2018)
17. Wu, L., Wang, Y., Ge, Z., Hu, Q., Li, X.: Structured deep hashing with convolutional neural networks for fast person re-identification. Comput. Vis. Image Underst. **167**, 63–73 (2018)
18. Wu, L., Wang, Y., Li, X., et al.: Deep attention-based spatially recursive networks for fine-grained visual recognition. IEEE Trans. Cybern. **PP**(99), 1–12 (2018)
19. Wu, L., Wang, Y., Li, X., Gao, J.: What-and-where to match: deep spatially multiplicative integration networks for person re-identification. Pattern Recognit. **76**, 727–738 (2018)
20. Yang, D., Shen, C., Lee, W., Chen, M.: On socio-spatial group query for location-based social networks. In: 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2012, Beijing, China, 12–16 August 2012, pp. 949–957 (2012)
21. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: efficient top K spatial keyword search. In: 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, 8–12 April 2013, pp. 901–912 (2013)
22. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: efficient top K spatial keyword search. IEEE Trans. Knowl. Data Eng. **28**(7), 1706–1721 (2016)
23. Zhang, C., Zhang, Y., Zhang, W., Lin, X., Cheema, M.A., Wang, X.: Diversified spatial keyword search on road networks. In: Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, 24–28 March 2014, pp. 367–378 (2014)
24. Zhang, D., Chee, Y.M., Mondal, A., Tung, A.K.H., Kitsuregawa, M.: Keyword search in spatial databases: towards searching by document. In: Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, 29 March–2 April 2009, Shanghai, China, pp. 688–699 (2009)